Local Search Heuristic Inspired in Particle Swarm Optimization for Reducing Collisions in Constrained Navigation (RCCN) Problems

Oregel J., Puga H., Zamudio V., Carpio M., Ornelas M. and Mancilla E.

Divsión de Estudios de Posgrado e Investigación
Instituto Tecnológico de León
León, Guanajuato, 37290, México
{josue-oregel, hector.puga}@itleon.edu.mx

Abstract: Collisions between *mobile objects* cause a negative impact to navigation systems, therefore reducing the number of collisions is deemed as an optimization problem. Besides, *Particle Swarm Optimization* is an algorithm intended to find solutions to optimization problems. This paper introduces a heuristic inspired in Particle Swarm Optimization to be applied to problems that we have called *Reducing Collisions in Constrained Navigation (RCCN)*, detecting collisions between mobile objects in a roadmap and reducing the number of collisions through the establishment of wait times. An analysis based on the outcome of the heuristic's application to an instance of RCCN problems is presented finding that is possible to reduce collisions between mobile objects using *Artificial Intelligence* techniques.

Key Words: Navigation, Optimization, Mobile Objects, Particle Swarm Optimization.

1. INTRODUCTION

The main task of an *Autonomous Mobile Object (AMO)* is to unwind itself on an environment in a natural way to achieve assigned tasks avoiding as possible problems, conflicts or complications in navigation. When the navigation system is bigger, the complexity increases and navigation planning can be considered as an optimization problem, for example for finding the shortest distance or movement time or taking out conflicts like collisions, deadlocks and congestions. For this reason it is desirable that AMOs can possess some kind of intelligence.

Artificial Intelligence (AI) can be defined as those systems that receive perceptions from the environment and react performing some actions in order to autonomously solve problems that usually require certain level of intelligence (Russell, et al., 2009). AI is a research area which focuses primarily in mimic fundamental aspects of biological intelligence to solve complex problems, displaying cognitive skills such as behavioral autonomy, social interaction, learning capabilities and evolution.

Particle Swarm Optimization (PSO) is a Bio-inspired optimization method which mimics the natural self-organizing behavior observed in many animal social groups, normally controlled by a leader, where individuals may not have total knowledge neither of the behavior of the group nor the environment. Despite this, they have the ability to gather and move together, based on local interactions between individuals (Floreano and Mattiussi, 2008). From the simple local interaction between individuals more complex collective behavior emerges. PSO was originally intended to find solutions in continuous spaces; however, some binarized versions have been developed in order to solve discrete problems (Engelbrecht, 2005).

A numerous quantity of research proposals related to autonomous navigation using AI techniques can be found in the state of the art with a diverse set of approaches like: agents, cell decomposition, flocking,

networking and robot perception, etc. (e.g. Chiew and Qin, 2009; Gendreau, et al., 2008; Ghasemzadeh, et al., 2009; Kala, et al., 2010; Lin, 2011; Mirtich, 1997; Peasgood, et al., 2008; Peng and Akella, 2005; Qiu and Hsu, 2003; Sharma, et al., 2005; Sujit, et al., 2012; Törnquist Krasemann, 2011).

In this work is addressed a set of problems that we have called *Reducing Collisions in Constrained Navigation (RCCN)*, which are related with known routing and scheduling in navigation problems as *Vehicle Routing Problems* (*VRP*) and *Automated Guided Vehicles* (*AGV*), where RCCN is different to both problems due to its characteristics.

The main contribution of this paper is the introduction of an optimization approach inspired in PSO to be applied in RCCN problems with the goal of detecting and reducing collisions between mobile objects on the intersections in constrained navigation systems with roadmaps given as graphs, where the nodes and edges connecting the graph can play the role of paths that are navigable by the mobile objects. The optimization's objective is achieved using a model that makes possible to detect collisions between the mobile objects in common nodes of the navigation system and provides a platform to reduce the number of collisions through the establishment of wait times in previous nodes.

This document is organized as follows: section 2 presents a briefly description of the RCCN problems and the solution model proposed, section 3 introduces a heuristic inspired in PSO to detect and reduce collisions in RCCN problems, in section 4, the application of the heuristic to a RCCN specific case is presented and section 5 displays the resulting outcome from the PSO approach application. Finally in section 6, a conclusion and work that is being done is discussed.

2. THE REDUCING COLLISIONS IN CONSTRAINED NAVIGATION (RCCN) PROBLEMS AND SOLUTION MODEL

These constraints define the RCCN problems:

1. The navigation environment is a roadmap considered as an undirected graph that connects its vertices (nodes) with navigable edges.
2. A path is a navigable by mobile objects succession of edges connected by nodes of the graph.
3. A vertice can belong to more than one navigation path.
4. An edge belongs to no more than one path and it can have just one navigation way.
5. Paths are predefined and immovable for each and all of the mobile objects within the navigation system.
6. A path is navigable only by one mobile object and it belongs only to one path, therefore, there are as many numbers of mobile objects as the number of paths in the system.
7. For study purposes, mobile objects are considered punctual elements, i.e., without dimensions.
8. The movement velocity is the same for all the mobile objects in the paths; it is constant and it keeps discreetly measured values.
9. A collision event occurs if there is a match at the same time between mobile objects in the same vertice of the roadmap; therefore, it is not allowed to accommodate more than one mobile object in a vertice at a time, except in the case of starting or ending vertices of a path, similar to what happens in real terminals.
10. It is possible to set a timeout or wait time ($te \mid te \in \mathbb{N} \cap 0$) for each mobile object at each vertice of its path, and this wait time can be changed at discrete intervals of time ($\Delta te \in \mathbb{N}$), in order to avoid possible collisions in common vertices of their path with paths of other mobile objects.
11. The beginning of the movement is simultaneous for all the mobile objects of the navigation system, i.e., there aren't allowed wait times for the mobile objects in initial vertices of their paths.

The total occupancy time of each node of a path is formed with the composition of three different data: the original movement time (inherited attribute), the wait time (assigned attribute) and the delay time caused by the wait times at preceding nodes in the path (calculated attribute).

The sum of the original movement time and the delay time caused by the accumulation of wait times in preceding nodes produce the arrival time to a node. Additionally, when a mobile object arrives to a node, it stays in the node for the interval of time earmarked, i.e., since a mobile object arrives to a node of its path and during the wait time assigned in the node, that mobile object will be occupying this node.

When a node common to two paths is not the initial or final node of one of the paths, a collision event is identified if there is an intersection between the intervals of occupancy of the node. To compute the

total collisions number, the collision events currently in the system are summarized.

Once a collision event has been detected, a wait time can be assigned for one of the conflictive paths in the preceding node to the conflicting common node in order to eliminate the collision event individually.

Even though it is possible to assign arbitrary wait time values to reduce collisions in the system, an implied intention in all the navigation systems is the accomplishment of all the paths in the shortest possible time, therefore is always sought the assignment of wait time values near to *0* for all the nodes of the paths.

3. HEURISTIC INSPIRED IN PSO

PSO (Kennedy and Eberhart, 1995) has a set of entities called particles in a population emulating the social behavior of natural social models, where the individuals of the population in a swarm can combine their own current history (φ_1) and global information (φ_2) to determine their movement through a search space based on an inertia factor (ω), looking for the solution to a problem (Algorithm 1).

Algorithm 1. PSO (taken from (Thiem and Lässig, 2011))

Input: Problem P and a solution space S
Output: Solution $\gamma \in S$
Requirement: ω:Velocity's influence factor
Requirement: φ_1: Particle's cognitive parameter
Requirement: φ_2: Particle's social parameter
1: $\boldsymbol{\alpha} \leftarrow$ compute_initial_states (*variables*)
2: $\boldsymbol{\beta} \leftarrow \boldsymbol{\alpha}$
3: **for** (all particles i from 1 to n) **do**
4: **if** (γ = null or $f(\beta_i) < f(\gamma)$) **then**
5: $\gamma \leftarrow \beta_i$
6: **end if**
7: **end for**
8: $\boldsymbol{\delta} \leftarrow \mathbf{0}$
9: **repeat**
10: $\boldsymbol{\alpha}, \boldsymbol{\delta} \leftarrow$ compute_new_state($\boldsymbol{\alpha},\boldsymbol{\beta},\gamma,\boldsymbol{\delta},\omega,\varphi_1,\varphi_2$)
11: **for** (all particles i from 1 to n) **do**
12: **if** ($f(\alpha_i) < f(\beta_i)$) **then**
13: $\beta_i \leftarrow \alpha_i$
14: **if** ($f(\beta_i) < f(\gamma)$) **then**
15: $\gamma \leftarrow \beta_i$
16: **end if**
17: **end if**
18: **end for**
19: **until** (termination_condition(variables))
20: **return** γ

Although PSO was proposed to solve continuous problems, several PSO versions have been developed to be applied on discrete problems (e.g. (Carlisle and Dozier, 2001; Clerc, 1999; Kennedy and Eberhart, 1997)).

3.1. The problem and the solution space

In RCCN problems are used two data sets to describe a navigation system:

1. A "Weighted Connectivity Matrix" (**Mc**), of *n x n* size, which is a weighted adjacency matrix and can be represented by a graph. The **Mc** matrix contains information about connections and movement time ($t_{(i\ j)}$) among the *n* nodes of the navigation system.

2. A set of sequences of nodes that represent a roadmap, i.e., each of the *m* navigation system paths ($\mathbf{R}_k$), where each path has its own quantity *s* of nodes (*n*) (equation 1).

$$\mathbf{R}_k = n_{(k\ 1)}, n_{(k\ 2)}, \ldots, n_{(k\ (s-1))}, n_{(k\ s)} \quad \text{where } 1 \le k \le m;\ k,m,s \in \mathbb{N} \tag{1}$$

To handle the total movement time to the paths, as well as to detect and to reduce collisions in the navigation system, three matrices are used: the **MTo** matrix to the original movement times, the **MTe** matrix to the wait times and the **MTr** matrix to the delay times, where solely the wait time's matrix **MTe** can be directly modified to manage the reduction of the collisions.

Once a modification has been made to **MTe**, it is necessary to compute the *problem's fitness function* value, i.e., the number of collisions in the navigation system. Each evaluation of the problem's fitness function is known here as a *function's call*.

3.2. Setting up initial states

Each one of the *n* particles α of the swarm has an initial position appointed by its position property, which it is an **MTe** matrix initialized with zero in all its positions (equation 2). Due to this fact, all the particles of the swarm have initially the same position, i.e., the same initial number of collision events.

$$\mathbf{MTe}_\alpha = \{te_{(i,j)}\}_\alpha \quad \text{where } te_{(i,j)}=0;\ 1 \le i \le m;\ 1 \le j \le n \tag{2}$$

3.3. How to Compute a New State

To compute a new particle's position implies to change the values of its **MTe** matrix elements, trying to reduce the number of collision events in the navigation system based on the model for RCCN problems proposed. The new particle's position is reached executing a random search of a node with a collision event, and once it has been found, randomly choosing one of the paths involved in the collision event and changing the wait time in the node located before of the conflictive node in the chosen path; i.e., changing the wait time in an element of the particle's **MTe** matrix supported by a random path selection criteria. It helps to make a binary decision to modify either the first or the second conflictive path (Algorithm. 2).

Algorithm 2. PSO's new state procedure

Requirement: A particle's wait times **MTe** matrix
1: **repeat**
2: randomly select a node from **MTe**
3: **if** (there is a collision in the node selected) **then**
4: randomly select a conflictive path
5: **if** (first conflictive path is selected) **then**
6: increase the wait time to the preceding node at the first path to the node selected
7: **else**
8: increase the wait time to the preceding node at the second path to the node selected
9: **end if**
10: **end if**
11: **until** (a collision event has been eliminated)
12: **return** the particle's **MTe** matrix updated

In this way, the position of the particles at each iteration of the heuristic would have a probabilistic chance of being modified (Cervantes, et al., 2005; Poli, et al., 2007) in either the first or the second path, conducting to reach particle's new positions in a stochastic way which produces different problem's fitness function values to the particles of the swarm; hence, there can be multiple wait time's configurations to reduce collisions in a navigation system.

3.4. Termination Criteria

There are two criteria to stop the heuristic:

1. When a specific number of function calls have been reached.

2. When there are not collisions in the navigation system.

4. APPLICATION OF THE PSO BASED HEURISTIC TO AN RCCN PROBLEMS INSTANCE

To test the PSO approach solving RCCN problems, a Java application was implemented and applied to a RCCN problems instance with a roadmap with 27 nodes and 5 routes where there are 6 collision events (Fig. 1).

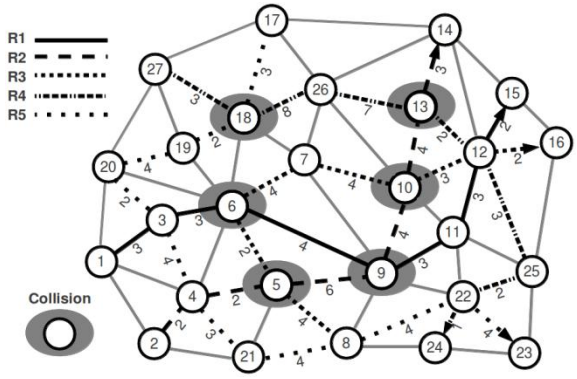

Fig. 1. An RCCN problems instance with 27 nodes, 5 navigation paths, and 6 collision events.

5. RESULTS

The PSO inspired heuristic applied to the RCCN problems instance using the number of function's calls as the termination criteria were executed with 6 different termination criteria values to observe its performance (Fig. 2, Table 1).

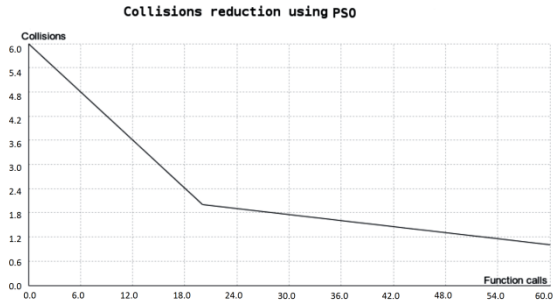

Fig. 2. PSO inspired heuristic's behavior using a RCCN problems instance.

Table 1. Results of PSO inspired heuristic using a RCCN problems instance

Termination criteria (function's calls)	Best problem's fitness function value (number of collisions)
0	6
10	6
20	3
30	2
40	1
60	1

Table 1 depicts that once the fitness function evaluation has been executed 40 or 60 times, the navigation system remains with a collision event; this is due to the restriction that indicates no wait time at the first node of a path. However, the collision event in node 18 between paths 4 and 5 cannot be eliminated.

A configuration to the RCCN problems instance presented here is described below: the particle's **MTe** matrix with the best problem's fitness function value determines the wait times to avoid collision events displayed in Table 2. Be aware that the path $\mathbf{R}_1$ does not change its wait times, i.e., its movement time remains unchanged.

Table 2. Wait times to the RCCN problems instance determined using the PSO inspired heuristic

Path	Configuration	
	Node	Wait time
$\mathbf{R}_2$	9	2
$\mathbf{R}_3$	8	1
$\mathbf{R}_4$	27	1
$\mathbf{R}_5$	8	1

6. CONCLUSIONS

Due to the fact that reducing the amount of collisions between *mobile objects* in navigation systems can be deemed as an optimization problem it is necessary to establish methodologies to try to solve this kind of problematic.

In this paper is presented a local search heuristic inspired in *PSO* to be applied to *Reducing Collisions in Constrained Navigation (RCCN)* problems with the goal to detect and reduce collisions in navigation systems, generating wait times in previous nodes to those which have collision events in the intersection of paths for mobile objects.

As a consequence of the application of the heuristic proposed to an instance of RCCN problems, it is observable that the heuristic is on average reducing collisions, i.e., each of the heuristic executions provides a solution result with different wait times configurations for the navigation system.

Actually we are working on generating test instances with a higher number of collision events in order to be able to implement the heuristic inspired on PSO using these test instances to obtain the average performance of the heuristic. The performance of the heuristic results provides the elements to compare it versus other techniques.

ACKNOWLEDGMENTS

This research was supported by the Consejo Nacional de Ciecia y Tecnología (CONACYT) and the Dirección General de Educación Superior Tecnológica (DGEST), project number 4572.12-P. Authors thank by the grammar check to Jose Urbina from Instituto Tecnológico de León.

REFERENCES

Carlisle, A. and Dozier, G. (2001). An Off-The-Shelf PSO. In *Proceedings of The Workshop On Particle Swarm Optimization*.

Cervantes, A., Galvan, I., and Isasi, P. (2005). A Comparison Between the Pittsburgh and Michigan Approaches for the Binary PSO Algorithm. In *Evolutionary Computation*, 2005. The 2005 IEEE Congress on, volume 1, pages 290-297. IEEE.

Chiew, K. and Qiu, S. (2009). Scheduling and Routing of AMOs in an Intelligent Transport System. *IEEE Transactions on Intelligent Transportation Systems*, 10-3:547-552.

Clerc, M. (1999). The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization. In *Evolutionary Computation*, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 3. IEEE.

Engelbrecht, A. P. (2005). *Fundamentals of Computational Swarm Intelligence*, volume 1. Wiley London.

Floreano, D. and Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. Intelligent Robotics and Autonomous Agents. The MIT Press, 1 edition.

Gendreau, M., Potvin, J., Brumlaysy, O., Hasle, G., and Lkketangen, A. (2008). Metaheuristics for the Vehicle Routing Problem and its Extensions: A Categorized Bibliography. *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 143-169.

Ghasemzadeh, H., Behrangi, E., and Abdollahi, A. M. (2009). Con
ict-free Scheduling and Routing of Automated Guided Vehicles in Grid Topologies. *Robotics and Autonomous Systems*, 57(6-7):738-748.

Kala, R., Shukla, A., and Tiwari, R. (2010). Fusion of Probabilistic A* Algorithm and Fuzzy Inference System for Robotic Path Planning. *Artificial Intelligence Review*, 33(4):307-327.

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. In *Neural Networks*, 1995. Proceedings., IEEE International Conference on, volume 4, pages 1942-1948. IEEE.

Kennedy, J. and Eberhart, R. (1997). A Discrete Binary Version of the Particle Swarm Algorithm. In *Systems, Man, and Cybernetics*, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, volume 5, pages 4104-4108. IEEE.

Lin, C. T. (2011). *Study on Vehicle Routing Problems With Time Windows Based on Enhanced Particle Swarm Optimization Approach*. Elsevier.

Mirtich, B. (1997). Efficient Algorithms for Two-Phase Collision Detection. *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, pages 203-223.

Peasgood, M., Clark, C., and McPhee, J. (2008). A Complete and Scalable Strategy for Coordinating Multiple Robots Within Roadmaps. *Robotics*, IEEE Transactions on, 24(2):283-292.

Peng, J. and Akella, S. (2005). Coordinating Multiple Robots With Kinodynamic Constraints Along Specied Paths. *The International Journal of Robotics Research*, 24(4):295-310.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle Swarm Optimization. *Swarm Intelligence*, 1:33-57.

Qiu, L. and Hsu, W. (2003). Continuous Scheduling of AGVs in a Grid-Like Path Topology. In *Intelligent Vehicles Symposium*, 2003. Proceedings. IEEE, pages 62-67. IEEE.

Russell, S. J., Norvig, P., and Davis, E. (2009). *Articial Intelligence: a Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 3 edition.

Sharma, V., Savchenko, M., Frazzoli, E., and Voulgaris, P. (2005). Time Complexity of Sensor-Based Vehicle Routing. In *Robotics: Science and Systems*, pages 297-304. Citeseer.

Sujit, P. B., Lucani, D. E., and Sousa, J. B. (2012). Bridging Cooperative Sensing and Route Planning of Autonomous Vehicles. *IEEE Journal on Selected Areas in Communications*, 30 No. 5:912-923.

Thiem, S. and Lässig, J. (2011). Comparative Study of Different Approaches to Particle Swarm Optimization in Theory and Practice. In *Particle Swarm Optimization*. Theory, Techniques and Applications, chapter 7, pages 127-167. Olsson, A. E., editor, Nova Science Publishers, Inc.

Törnquist Krasemann, J. (2011). Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances. *Transportation Research Part C: Emerging Technologies*.