Marius M. BĂLAŞ
„Aurel Vlaicu" University of Arad,
Engineering Faculty
Bd. Revoluției nr. 77, 310130, Arad,
Romania
E-mail: marius.balas@ieee.org

Valentina E. BĂLAŞ
„Aurel Vlaicu" University of Arad,
Engineering Faculty
Bd. Revoluției nr. 77, 310130, Arad,
Romania
E-mail: balas@inext.ro

APPLYING THE CONSTANT TIME TO COLLISION CRITERION IN SWARM SYSTEMS

NOTE: *This paper was presented at the International Symposium "Research and Education in an Innovation Era", Engineering Sciences, November 20-21, 2008, "Aurel Vlaicu" University of Arad, Romania*

ABSTRACT:

The paper is making a short introduction into the field of the swarm intelligent robots and is proposing a new approach for the self-organizing swarms, based on the criterion of the constant time to collision. This criterion is imposing an optimal distance between moving particles, such way that the times to collision between particles are constant, for any speed. The same time to collision is imposed to the whole swarm. The imposed time to collision and therefore the distance gaps between the particles can be adjusted. Such way each member of a moving swarm can find by itself a position that is optimizing the structure and the dimensions of the swarm, according to its speed. A simulation is provided for a simple case: the Indian run.

KEYWORDS:

Swarm intelligent algorithms, particle swarm optimization, constant time to collision criterion.

INTRODUCTION. THE SWARM INTELLIGENCE

The Swarm Intelligent paradigm (SI) [Kennedy and Eberhart 2001, Clerc 2006] is inspired from the social dynamics and emergent behavior that arise in socially organized colonies. The importance of such a concept in the control theory is linked to the idea of the colonies of robots [1], [2], etc. The aim is to replace an individual exploring, working or fighting robot (which is complicate, expensive, and exposed to different failure mechanisms) with a group of much smaller robots (simple, cheap, replaceable), that will act in a self-organized way, inspired by social behavior patterns of organisms that live and interact within large groups. A mathematical concept that is supporting this approach is the Particle Swarm Optimization algorithm (PSO), which may incorporate swarming behaviors observed to birds, fish, bees, ants and even human social behavior [1], [3], [4]. Our purpose is to introduce in swarm systems an optimization criterion that was previously used in the Automate Cruse Control: the Constant Time to Collision (CTTC).

AN INTRODUCTION INTO THE PARTICLE SWARM OPTIMIZATION

PSO is learning algorithm, exploiting a population of individuals to probe promising regions of the search space. In this context, the population is called *swarm* and the individuals are called *particles*. Each particle moves with an adaptable velocity within the search space, and retains a memory of the best position it ever encountered. In the *global* variant of PSO, the best position ever attained by all individuals of the swarm is communicated to all the particles. In the *local* variant, each particle is assigned to a

topological neighborhood consisting of a prespecified number of particles. In this case, the best position ever attained by the particles that comprise the neighborhood is communicated among them [4].

Assume a D–dimensional search space, $S \subset \Re^D$, and a swarm consisting of N particles. The i–th particle is in effect a D–dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^T \in S$. The velocity of this particle is also a D–dimensional vector, $Vi = (vi1, vi2, \ldots, viD)^T \in S$. The best previous position encountered by the i–th particle is a point in S, denoted by $P_i = (P_{i1}, P_{i2}, \ldots, P_{iD})^T \in S$. Assume gi to be the index of the particle that attained the best previous position among all the particles in the neighborhood of the i–th particle, and t the iteration counter. Then, the swarm is manipulated by the following equations [5]:

$$V_i(t + 1) = \chi [wV_i(t) + c_1 r_1 (P_i(t) - Xi(t)) + c2\ r2\ (P_{gi}(t) - Xi(t))], \quad (1)$$

$$X_i(t + 1) = X_i(t) + V_i(t + 1), \quad (2)$$

where $i = 1, \ldots, N$; $c1$ and $c2$ are two parameters called *cognitive* and *social* parameters respectively; $r1$, $r2$, are random numbers uniformly distributed within [0, 1], and gi is the index of the particle that attained either the best position of the whole swarm (global version), or the best position in the neighborhood of the i–th particle (local version). The parameters χ and w are called *constriction factor* and *inertia weight* respectively, and they are used as mechanisms for the control of the velocity's magnitude, corresponding to the two main PSO versions. The value of the constriction factor is derived analytically [5]. On the other hand, the inertia weight, w, is computed empirically, taking into consideration that large values encourage global exploration, while small values promote local exploration. According to a rule of thumb, an initial value of w around 1.0 and a gradual decline towards 0 is considered a proper choice. In general, the constriction factor version of PSO is faster than the one with the inertia weight, although in some applications its global variant suffers from premature convergence. Regarding the social and cognitive parameter, the default values $c1 = c2 = 2$ have been proposed. The

initialization of the swarm and the velocities is usually performed randomly and uniformly in the search space, although more sophisticated initialization techniques can enhance the overall performance of the algorithm [4].

THE CONSTANT TIME TO COLLISION CRITERION

As shown before, swarm systems may be abstractized and used in automate learning. In this paper we will come back to the original sense of the concept: swarm system = a group of interacting automobile objects. Our goal is to find an algorithm that is able to optimize the swarm movements, by minimizing the distance between individuals, with respect to a common collision risk. The first step in this direction is to investigate one of the simplest swarm models: the Indian run. The Indian run may be observed in nature at many species of ants, birds, or mammals (elephants for instance), as well as in different social activities, namely in sports: cycling, athletics, etc. We will associate the Indian run to a concept belonging to the Automate cruise control: the Constant Time to Collision (CTTC) criterion [6], [7]. TTC is the time before two following cars (Car2 is following Car1) are colliding, assuming unchanged speeds of both vehicles:

$$TTC = \frac{d_{21}}{v_2 - v_1} \tag{1}$$

where v_1 and v_2 are the speeds of the vehicles and d_{21} the distance gap between them.

CTTC consists in imposing *stabilized* TTCs by means of the Car2 cruise controller. The on-line TTC control is not convenient because when the two cars have the same speed the TTC's denominator is turning null: v2 - v1= 0. That is why CTTC must be implemented off-line, with the help of $d_i(v2)$ mappings (**fig. 1**). The CTTC implementation by $d_i(v_2)$ distance-gap planners is possible because *a distance gap planner using* TTC *will produce*

CTTC. We studied this method by computer simulations, using a Matlab-Simulink model of the tandem Car1-Car2 [6].

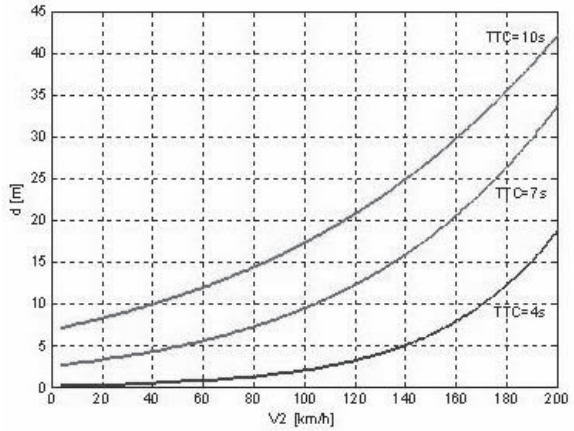

Fig. 1. $d_i(v_2)$ mappings for three different TTC

The distance-gap planners are designed by means of a computer simulation, as follows. The simulation scenario consists in braking Car1 until the car is immobilized, starting from a high initial speed. A TTC controller is driving the Car2 traction/braking force such way that during the whole simulation TTC is stabilized to a desired constant value. The continuous braking allow us to avoid the v_2-v_1=0 case. We will use the recorded d mapping as the desired $d_i(v_2)$ planner for the given TTC. The **Fig. 1** planners are determined for three TTC values: 4s, 7s and 10s. The **Fig. 2** is presenting the computer model.

Applying CTTC brings two obvious advantages:

- a constant collision risk for each vehicle involved;
- the possibility to control the traffic flow on extended road sections, if each vehicle will apply the same TTC that is currently recommended by the Traffic Management Center: a long TTC means *low traffic* flow and *higher safety* while a short TTC means *high traffic* flow and *higher risk*.

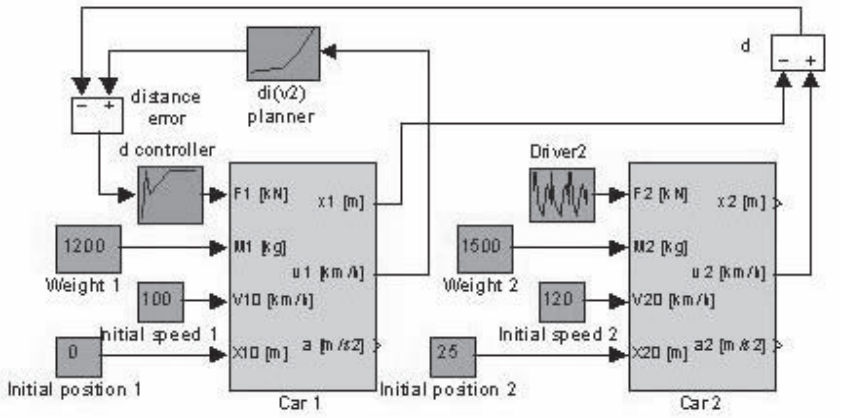

Fig. 2. A cruise control system with distance controller and CTTC $d_i(v_2)$ planner

A CTTC PLATOON SIMULATION

The following Simulink-Matlab model allows us to simulate the behavior of a CTTC platoon, running in Indian style (see Fig. 3). The elements of the platoon are five identical cars, the first one driven by a driver. Each car is provided with a PID distance controller that is following as close as possible the imposed distance di(v), and therefore the imposed TTC. The imposed TTC is variable: 10s for the first 200 seconds of the simulation and 7 s for the last 120 seconds, linked by a ramp transition. All the cars are starting from the same spot.

The speed of the first car, Car1, is presented in Fig. 4. The distances between the cars and the overall length of the platoon are presented in fig. 5. One can easily observe the continuous variation of the platoon's length:

a) with the speed (for the first 200s), and

b) with the imposed TTC (for the last 220s).

Fig. 6 is showing the initialization of the platoon, which is perfectible.

The simulation is illustrating the simplest case of a swarm movement: the longitudinal drive. The next stages of this research should extend the method for the 2D and 3D cases, and refine the control algorithms, that will improve the dynamics of the platoon.

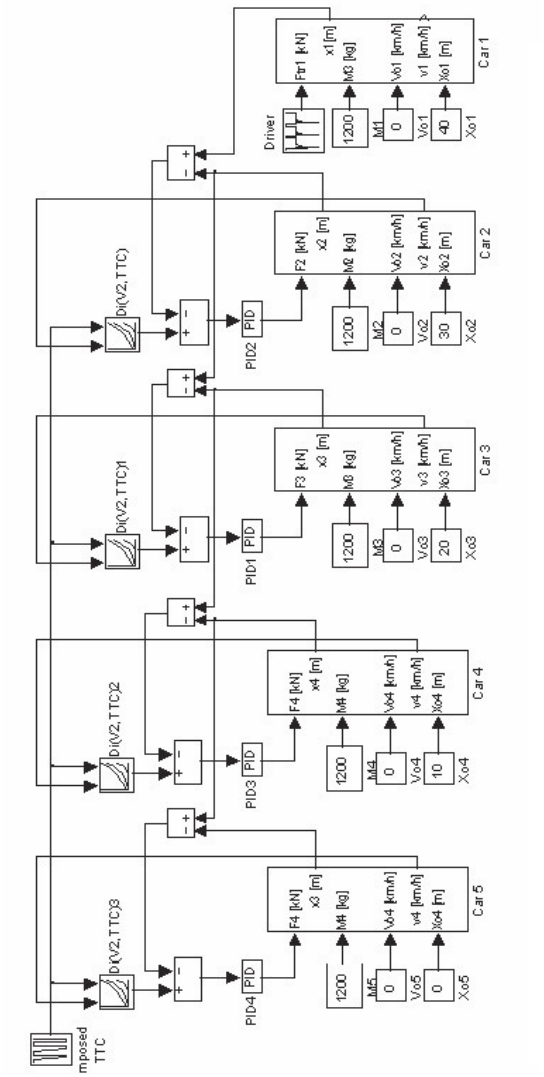

Fig. 3. The Simulink model of a 5 automobiles platoon

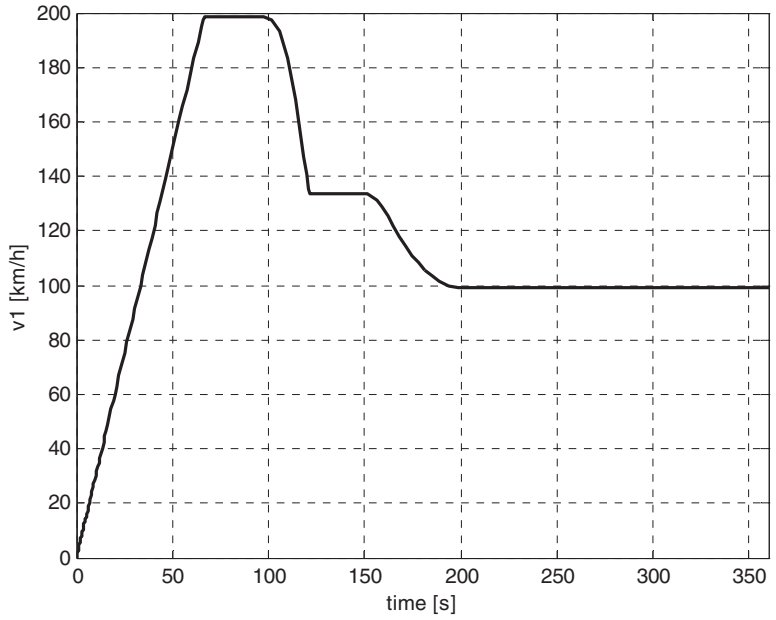

Fig. 4. The speed of the first car

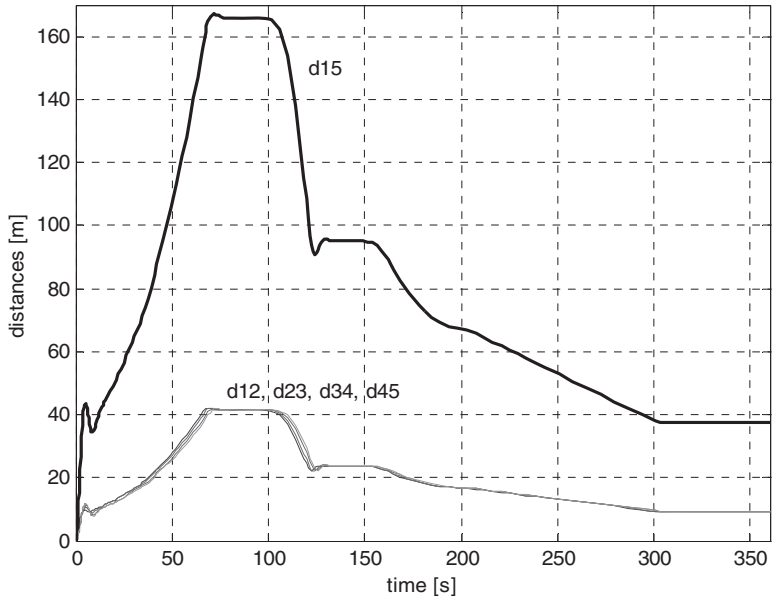

Fig. 5. The distances between cars and the length of the platoon

CONCLUSIONS

The constant time to collision criterion can stand for an optimization method for moving swarm systems. The particles must preserve an optimized distance with their neighbors, such way that the time to collision is constant and adjustable for all the swarm. The distances between particles are continuously adapted to the actual speed and the dimensions of the swarm are minimized. In the same time, the collision risk, that depends of the imposed time to collision is evenly distributed. The method is illustrated by a simulation of an 5 automobiles platoon.

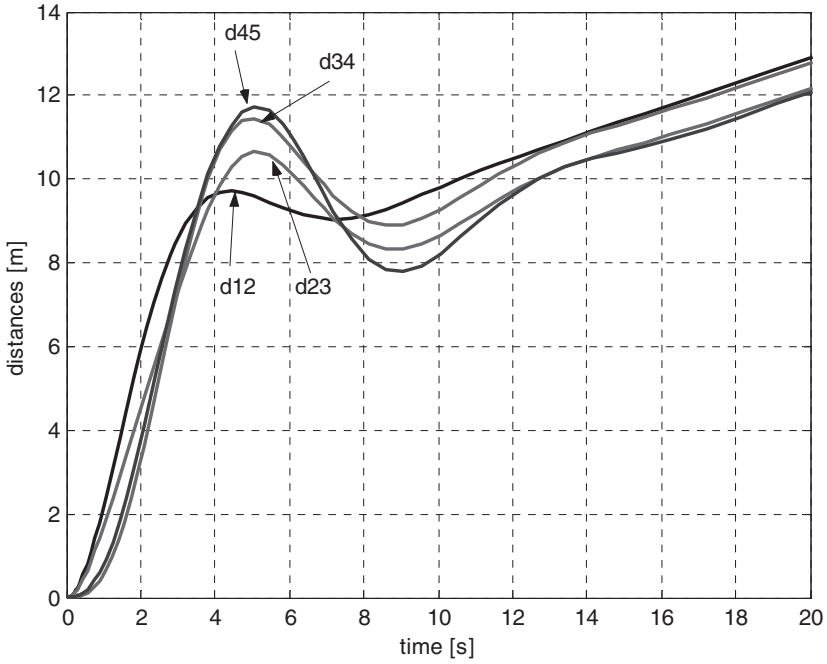

Fig. 6. The initialization of the platoon

REFERENCES

[1] G. Lefranc. Des colonies de robots : un nouveau défi. *Session plénière, Conférence Internationale Francophone d'Automatique CIFA 2008,* Bucarest, Septembre, 2008.
[2] J.C. Braly. The Development of a Low-Cost and Robust Autonomous Robot Colony Using LEGO® Mindstorms™. *MS thesis, North Carolina State University, Raleigh*, 2003.
[3] Hongbo Liu, Ajith Abraham. An Hybrid Fuzzy Variable Neighborhood Particle Swarm Optimization Algorithm for Solving Quadratic Assignment Problems. *Journal of Universal Computer Science*, vol. 13, no. 9 (2007), 1309-1331.
[4] K.E. Parsopoulos, E.I. Papageorgiou, P.P. Groumpos, M.N. Vrahatis**.** A First Study of Fuzzy Cognitive Maps Learning Using Particle Swarm Optimization. *Proc. of the IEEE 2003 Congress on Evolutionary Computation*, Canberra, http://www.math.upatras.gr/~kostasp/ papers/ cec03a.pdf.
[5] M. Clerc and J. Kennedy. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1), pp. 58–73, 2001.
[6] M. Balas, V. Balas, J. Duplaix. Optimizing the Distance-Gap between Cars by Constant Time to Collision Planning. *Proc. of IEEE International Symposium on Industrial Electronics ISIE 2007*, June 2007, Vigo, pp. 304-309.
[7] M.M. Balas, V.E. Balas. Constant Time to Collision Platoons. *International Journal of Computer Communications & Control,* ISSN 1841-9836, E-ISSN 1841-9844, vol. III (2008), Suppl. issue: Proceedings of ICCCC 2008, pp. 33-39, 15-17 Mai, 2008, Oradea, Romania.