DE GRUYTER OPEN

JAISCR, 2018, Vol. 8, No. 1, pp. 63 – 77
10.1515/jaiscr-2018-0004

USING PARTICLE SWARM OPTIMIZATION TO ACCURATELY IDENTIFY SYNTACTIC PHRASES IN FREE TEXT

George Tambouratzis

Dept. of Machine Translation, Institute for Language and Speech Processing / Athena Research Centre
6 Artemidos & Epidavrou Str., Paradissos Amaroussiou, 151 25, Greece

Submitted: 17th January 2017; accepted: 29th March 2017

Abstract

The present article reviews the application of Particle Swarm Optimization (PSO) algorithms to optimize a phrasing model, which splits any text into linguistically-motivated phrases. In terms of its functionality, this phrasing model is equivalent to a shallow parser. The phrasing model combines attractive and repulsive forces between neighbouring words in a sentence to determine which segmentation points are required. The extrapolation of phrases in the specific application is aimed towards the automatic translation of unconstrained text from a source language to a target language via a phrase-based system, and thus the phrasing needs to be accurate and consistent to the training data.

Experimental results indicate that PSO is effective in optimising the weights of the proposed parser system, using two different variants, namely sPSO and AdPSO. These variants result in statistically significant improvements over earlier phrasing results. An analysis of the experimental results leads to a proposed modification in the PSO algorithm, to prevent the swarm from stagnation, by improving the handling of the velocity component of particles. This modification results in more effective training sequences where the search for new solutions is extended in comparison to the basic PSO algorithm. As a consequence, further improvements are achieved in the accuracy of the phrasing module.

Keywords: parsing of natural language, machine translation, syntactically-derived phrasing, particle swarm optimization (PSO), parameter optimization, Adaptive PSO (AdPSO)

1 Introduction

The present work has been motivated by the need to implement Machine Translation systems. Machine Translation (MT) is a key task that provides humans with direct access via their own native language to textual data. MT is aimed to translate textual data written in a given language (termed the source language - SL) to a desired language (the target language - TL).

The MT task has been studied for over 50 years, but only with the advent of MT systems based on Machine Learning, it has become feasible to translate original textual content to unconstrained target languages, without having to handcraft a large set of translation rules. Statistical MT [1, 2] works by extracting statistical information from parallel sentences in the source and target languages, in order to create a translation model. The key SMT advantage is that it is directly amenable to new language pairs using a standard algorithmic suite. More recently, Neural MT (NMT) has been proposed [3], which uses multi-layer neural networks coupled with deep-learning techniques to establish translation systems. The drawback of both SMT and NMT is that they require voluminous parallel corpora (of the order of millions of words in both SL and TL) from which to extract the required translation models. Such parallel corpora may not be available but for a few SL/TL language pairs, limiting the applicability of these systems.

PRESEMT [4] was designed to make feasible the creation of MT systems with very small parallel corpora (of only a few hundred sentences) and with limited availability of linguistic tools for SL and/or TL. This allows the coverage of substantially more language pairs than SMT or NMT (mainly the lesser-resourced pairs), by virtue of the reduced requirement for parallel training data. The parallel corpus is supplemented by large amounts of monolingual corpora, to achieve a competitive translation quality without explicit provision of linguistic knowledge. The potential lack of tools for either SL or TL is addressed by developing algorithmic solutions that transfer information from SL to TL and vice versa.

When designing PRESEMT, it was decided to base the translation process on the handling of syntactically motivated phrases. During translation, each sentence is split into a sequence of phrases, which are then re-ordered and translated (for details cf. [4]). The correct identification of syntactic phrases within the pre-processing step is of importance to the generation of accurate translations. This process is generally fulfilled by a shallow parser, which establishes the syntactic phrase boundaries but not the deeper syntactic relations between phrases. In PRESEMT, to reduce the requirements for linguistic tools, a dedicated module (the Phrase Model Generator (PMG) module) is invoked – prior to the translation process – to identify the constituent phrases within the text being translated. The training data for PMG originates from the TL-side text that has been segmented into phrases by an existing linguistic tool. This phrasing is ported via the small parallel corpus to the SL-side, producing a segmentation into phrases. This segmentation is then used to train PMG, so that it becomes able to segment arbitrary SL-side text in agreement with the TL-side phrasing scheme. An example of this process is depicted in Figure 1.

In the present article, emphasis is placed on the optimization of a phrasing model, which is based on the principle of attraction and repulsion forces. The main aim is to determine the optimal parameters of this phrasing model via a particle swarm optimization (PSO) algorithm [5]. Modifications of the PSO algorithm are examined and the ensuing improvements to the segmentation quality are evaluated.

2 Creating a Phrasing Model with Limited Training Data

The phrasing model aims to extrapolate a matching phrasing model between two languages (the source and target languages) on the basis of a phrasing scheme on one side (TL). In the specific task studied here, a very limited amount of parallel training data is available. Thus a key constraint regards the data, on which to train the model, which covers typically only a two hundred sentences.

2.1 Existing Phrasing Models

Probably the most widely-used phrasing model implementation is via the stochastic model of Conditional Random Fields (CRF) [6]. CRF has a very powerful representation capability, when taking into account context (here context is expressed as the effect of neighbouring words and phrases on the type and boundaries of a phrase). CRF has been applied to language modelling by numerous researchers, including Sha et al. [7], Finkel et al. [8], Tsuruoka et al. [9], and Durrett et al. [10]. However, the high representation capability of the CRF mathematical modelling implies a large volume of training patterns to extract a model with sufficient performance. This violates the requirement of covering languages with limited resources, since in the setting studied here the training dataset comprises only a few hundred sentences.

A second phrasing model developed for PRESEMT is based on Template Matching (TEM), which adopts a learn-by-example approach. In TEM the phrasing model identifies phrases that precisely match previously encountered training patterns [11]. TEM matches parts of the sentence with a list of valid phrases that have been arranged in descending order of their likelihood of being correct. Thus the phrasing algorithm matches the most reliable phrases first. Experimental results [11] show that TEM attains a higher phrasing accuracy than CRF, learning more effectively from the small training set.

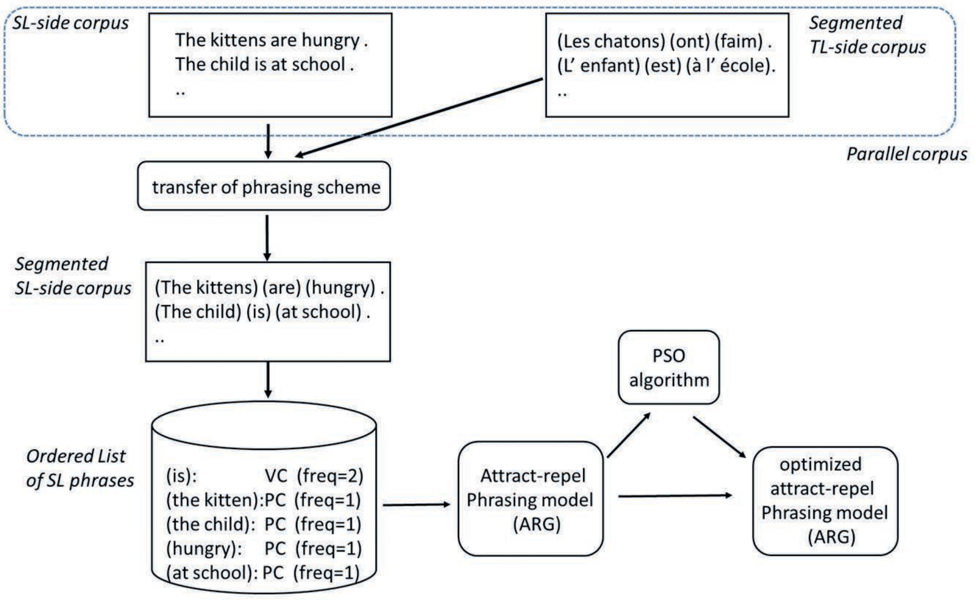


Figure 1. Schematic diagram indicating the development of the phrasing model based on a given training dataset

2.2 Definition of Attraction and Repulsion Forces

In analogy to the attraction-repulsion forces encountered in nature, many researchers have emulated the concept of attraction-repulsion to formulate artificial intelligence tasks. This formulation is combined with populations of agents, such as ant-colony optimization [12] and bee colony optimization [13]. Attractive and repulsive forces have been used by Yu et al. [14], for segmenting an image into different parts. Focusing on language-related applications, the concept of attraction-repulsion has been employed for phrasing purposes [15] in Text-to-Speech synthesis.

In present work, the task is to transfer an existing phrasing scheme from one language to another, by using a very limited training set. The requirement is for the SL phrasing to agree with the TL-side phrasing as closely as possible. Within the input text, at the point between any two consecutive words in a sentence, one needs to decide whether a phrase boundary is needed or not. This decision is taken by combining attractive and repulsive forces between words.

The definition of attractive and repulsive forces is introduced via an example in the following Section. The occurrence in the training set of word sequence $\{W_i\ ;\ W_{i+1}\}$ within one phrase represents a positive force between the two words that attracts them to each other, and thus tends to place them in the same phrase. On the contrary, the assignment of words W_i and W_{i+1} to different consecutive phrases represents a strong repulsive force, encouraging the addition of a phrase boundary between them. As pairs of words are studied, the range of this force is equal to 2. As a second example, the frequent termination of a phrase after W_i represents a repulsion force between W_i and its following word W_{i+1}, irrespective of the identity of W_{i+1}. In this case the force takes into account a single word (the sampled range of words being equal to 1).

Attractive and repulsive forces (represented as *F_attr(i)* and *F_repel(i)* respectively) are combined to give a constituent value, denoted as *Pot*, which corresponds to the potential of the specific point being a phrase boundary. For each possible segmentation point, depending on the chosen modelling, a number of attractive and repulsive forces are defined (denoted by index *j*), each of which samples a different range of words. *Pot* is defined as a linear combination of forces

$$Pot(i) = \sum_{j=1}^{k}(a_j \bullet F\_attr_j(i)) - \\ - \sum_{i=1}^{k}(b_j \bullet F\_repel_j(i)) - thres \quad (1)$$

The total number of parameters in (1) is equal to (*2k+1*). The actual set of attractive and repellent forces is enumerated in Table 1, for the configuration adopted in the present article, where forces are defined over word ranges from 1 to 4.

Table 1. Constituent Forces in Attract-Repel

Id.	**Type of Attractive and Repulsive Force**		
	Range (sampled words)	***Attractive force***	***Repulsive force***
1	1 (word w_i)	F_attr_1	F_repel_1
2	1 (word w_{i+1})	F_attr_2	F_repel_2
3	2 (words w_i & w_{i+1})	F_attr_3	F_repel_3
4	4 (w_{i-1}, w_i, w_{i+1}, w_{i+2})	F_attr_4	F_repel_4

The decision whether to place a phrase boundary at a given point, depends on the summation of constituent forces, each multiplied by a related weight. When the potential for a given sentence point is negative, it indicates that a phrase boundary is located at this point. The aim then becomes to determine the optimal weight parameters of the model, as listed in Table 1.

An example sentence is depicted in Figure 2. Based on the evidence provided by this simple sample sentence, the occurrence of the bigram "the child", where both words belong to the same phrase, would give a strong attractive force (F_attr_3)at the point just after word "the". Similarly, the repulsive force after the word "child" (F_repel_1) would be high (to indicate the increased likelihood of a segmentation point).

A training set of 200 sentences, with 5 to 10 phrases each, results in approximately 1,500 instances of phrases as training patterns. Such limited numbers of instances pose a data sparsity issue. To learn efficiently from such sparse data, the corresponding part-of-speech tags replace the words in the sentences (an example is shown in Figure 2). The introduction of tags results in the calculation of frequencies over higher-level features, allowing the system to learn more effectively.

Initial experiments with the phrase generator based on the Attract-Repel principle (hereafter this generator is denoted as ARG) have involved the manual setting of parameters, to identify whether this phrasing model can generate a competitive phrasing performance (cf. [16] for more details). Forces that operate over a longer range (and consequently necessitate the matching of a longer sequence of word tags) should have a higher multiplicative weight a_i (or b_i), to indicate the relative value of information derived from more complex patterns. This manual process does not guarantee the achievement of a global optimum. As an alternative, the systematic optimization of the phrasing model parameters via a PSO metaheuristic has been studied.

3 Summary of the Selected PSO Approach

One of the most widely-used metaheuristics for parameter optimization is particle swarm optimization (PSO) [5]. PSO replicates the collective behavior of a swarm of living organisms, by employing a set of simple particles, each of which searches for the best solution. During this search the particles interact with each other while traversing the pattern space. The next position to be evaluated by the l^{th} particle is defined by its current position $X_l(t)$, the best solution it has previously located $P_l(t)$, and the previous best solution $P_g(t)$ determined by all particles within its current neighbourhood. Depending on whether the local neighbourhood covers the entire swarm or a smaller region around the given particle, the swarm is termed as fully-informed or canonical, respectively [17].

In PSO, two variables define the search of a particle through the pattern space, namely (i) the particle's (i) location and (ii) velocity. The dimensionality of the two variables is equal to that of the pattern space. The new location and new velocity of the l^{th} particle at time step *t+1* are defined as

$$\begin{aligned} \vec{v_l}(t+1) = &\vec{v_l}(t) + \\ &+ \varphi_1 \vec{U}[0,\varphi_1] \bullet (P_l(t) - X_l(t)) + \\ &+ \varphi_2 \vec{U}[0,\varphi_2] \bullet (P_g(t) - X_l(t)), \end{aligned} \tag{2}$$

$$\vec{x_l}(t+1) = \vec{x_l}(t+1) + \vec{v_l}(t+1). \tag{3}$$

In equation (2), $U(y,z)$ is a vector of random numbers, each within a range between y and z. The PSO search can be distinguished into (i) exploration

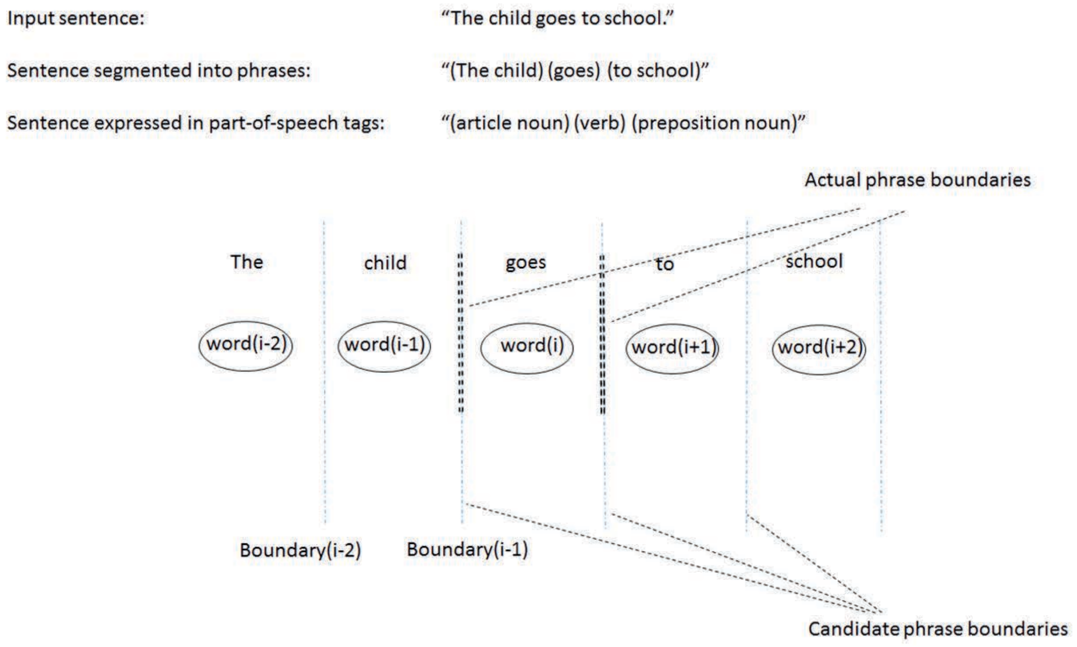

Input sentence:
"The child goes to school."
Sentence segmented into phrases:
"(The child) (goes) (to school)"
Sentence expressed in part-of-speech tags:
"(article noun) (verb) (preposition noun)"
Actual phrase boundaries
The
child
goes
to
school
word(i-2)
word(i-1)
word(i)
word(i+1)
word(i+2)
Boundary(i-2)
Boundary(i-1)
Candidate phrase boundaries

Figure 2. Example of sentence and its separation into syntactically annotated phrases. In the lower part of the figure, the candidate phrase boundaries are indicated by single dashed lines and actual boundaries are indicated by twin dashed lines.

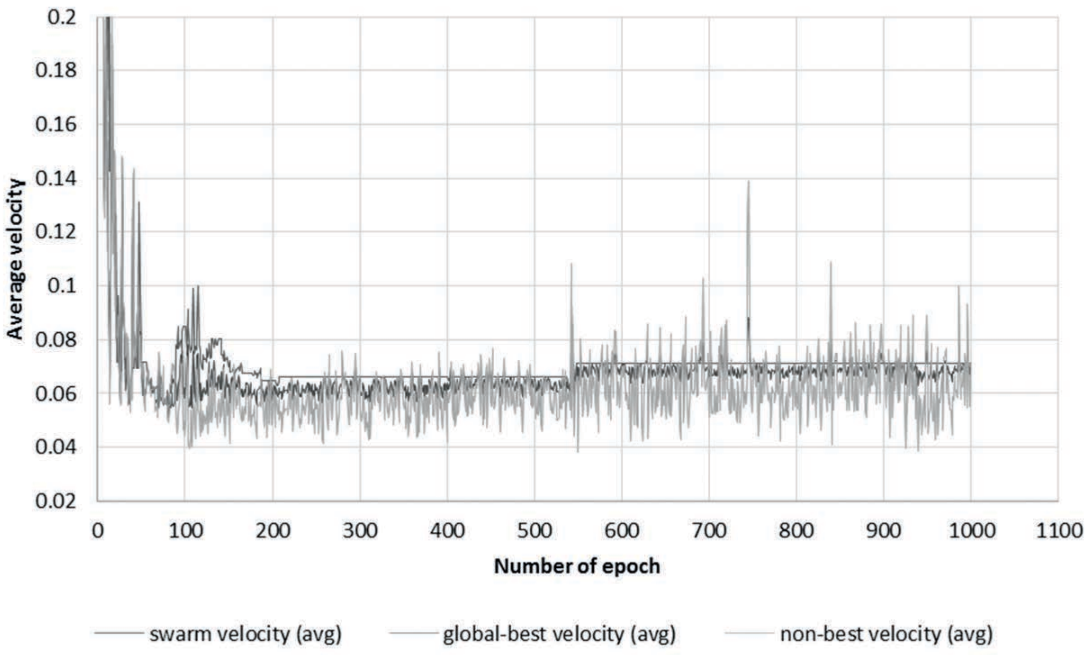


Figure 3. The average velocity of the swarm particles as well as the averages of velocities achiveing the global best solutions and not achieving the global best

(i.e. search of new portions of the pattern space) and (ii) exploitation (local search near discovered competitive solutions). Efforts have been made to balance PSO exploration and exploitation [18] and also to increase the exploration capability of PSO variants [19]. A basic PSO algorithm has been applied to the task of ARG weight optimization in a preliminary study [16]. In the present article, results on new experiments are presented, using two PSO variants that are comparatively investigated. The present article builds on the earlier work to improve the handling of velocity within the PSO evolution phase.

The first PSO variant is denoted sPSO (for standard-PSO) and has originated as a direct implementation (by the authors) of the basic PSO algorithm [5]. sPSO augments the basic PSO with (a) an elitism component (the best solution located so far is retained in all subsequent epochs), as well as (b) a local-search process. Regarding the latter component, we have adopted the approach of Gimmler et al. [20], using Powell's iterative method [21] to determine alternative local solutions with the algorithm of Press et al. [22]. For sPSO, φ_1 and φ_2 (cf. equation (2)) take the values of 0.729 and 4.100 respectively, in accordance to [23].

The second PSO variant is Advanced PSO (hereafter referred to as AdPSO) [24]. AdPSO has been reported to be more effective than sPSO over a set of standard test functions [24]. The AdPSO algorithm has been reimplemented from scratch for the research reported here, adhering to Zhan et al. [24]. In AdPSO, four different states of search for solutions are defined. The values of parameters φ_1 and φ_2 are algorithmically varied during run-time, depending on the current search state, enabling the swarm to switch between exploration and exploitation as appropriate. According to Zhan et al. [24], AdPSO does not require a local search component, in contrast to sPSO. Elitism is an integral part of AdPSO, and is coupled with a perturbation algorithm that exploits the search space around the best solution $P_g(t)$, to find better local solutions.

4 Experimental Setup and Results

In our experiments, two swarm sizes are used, the larger one comprising 80 particles (arranged in a 8x10 structure) and the smaller one 20 particles (in a 4x5 structure). Each swarm is allowed to evolve for 1,000 epochs. For each configuration tested, 10 independent runs have been implemented, to determine more accurately the characteristics of the solutions. Local search is invoked in sPSO every 50 epochs, using a rate similar to other researchers.

4.1 Measuring the Segmentation Accuracy

Segmentation accuracy is determined as the percentage of discovered segmentation points that match the ideal segmentation points between phrases for a test set. An automatically-calculated metric is required to guide PSO during the evolution of the swarm, at each point estimating the quality of the current solution. The metrics used here are based on the definitions of *tp* (true positives), *tn* (true negatives), *fp* (false positives) and *fn* (false negatives). Both the positive and negative instances are taken into account to reach an accurate segmentation, leading to the Precision and Recall metrics as defined in (4) and (5). From these, the F-Measure, Accuracy and F_2-measure metrics are defined from equations (6), (7) and (8), respectively.

$$precision = \frac{tp}{tp+fp}, \tag{4}$$

$$recall = \frac{tp}{tp+fn}, \tag{5}$$

$$Accuracy = \frac{tp+tn}{tp+fp+tn+fn}, \tag{6}$$

$$F-measure = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \tag{7}$$

$$F_b-measure = (1+b^2) \cdot \frac{precision \cdot recall}{b^2 \cdot precision + recall}. \tag{8}$$

4.2 Manually Setting the Attraction-Repulsion Parameters

Initially, the set of Attraction-Repulsion parameters was determined manually, by implementing a series of experimental runs. To simplify the selection of weights, the configurations tested assume symmetric values of the parameters (i.e. weights a_i and b_i for attractive and repellent forces were given the same values for the same range). The best manually-defined configuration is depicted in Table 2 with the corresponding scores achieved.

Table 2. Optimal ARG Configuration with Manually Defined Weights and corrsponding Phrasing Accuracy

Weights organized by type			
Range 1	***Range 2***	***Range 4***	***Thres***
$a_1 = a_2$=*0.01* $b_1 = b_2$=*0.01*	$a_3 = b_3$ =*0.10*	$a_4 = b_4$ =*1.00*	*0*
Phrasing Accuracy			
Precision	Recall	F-measure	
0.9790	0.9590	0.9389	

Table 3. Translation Quality Metrics Achieved by Different Phrasing Models. The Improvement over the Baseline (CRF) is Denoted in Brackets as a Percentage

Phrasing algorithm	**Translation quality**		
	BLEU	***NIST***	***Notes***
CRF	0.3289	6.8347	*Baseline solution*
TEM	0.3667 *(+11.49%)*	7.1372 *(+4.43%)*	*Optimum solution*
ARG	0.3515 *(+6.87%)*	*7.0613 (+3.32%)*	*Best manually-tuned solution*

Table 4. F-measure Obtained with ARG via PSO and AdPSO (80 particles), in Terms of the Best Solution (minimum) and the Average over 10 Runs

Swarm type	**F-measure scores (over 10 runs)**			
	Best solution		***Average***	
	F-measure	***epoch***	***F-measure***	***epoch***
sPSO with local search	0.94458	638	0.94167	173.4
AdPSO	0.94306	452	0.94107	124.6

Table 5. F-measure Obtained with ARG via PSO and AdPSO (20 Particles), in Terms of the Best Solution and the Average over 10 Runs

Swarm type	**F-measure scores over 10 runs**			
	Best solution		***Average***	
	F-measure	***epoch***	***F-measure***	***epoch***
sPSO with local search	0.92406	142	0.90939	73.5
sPSO without local search	0.92391	206	0.90890	129.2
AdPSO	0.91535	858	0.90351	257.6

To determine how ARG compares to alternative phrasing modules, it was integrated to the PRESEMT translation system. At this point, the manually-tuned version of Table 1 is used. The MT system used for experimentation involves translating from the Greek language (SL) to the English (TL) one, using the configuration discussed in [11]. The default parallel corpus of PRESEMT has been used (available at www.presemt.eu), which comprises 200 parallel sentences. The translation performance is evaluated on the PRESEMT test corpora (available at www.presemt.eu). The TreeTagger tool ([25], [26]) processes the TL side of the parallel corpus to determine the part-of-speech tags of words and divide the sentences into phrases. The corresponding phrasing in SL was determined using the PRESEMT phrase-alignment module [4]. The results using two widely-used objective measures for translation tasks, namely BLEU [27] and NIST [28], are shown in Table 3. For both BLEU and NIST a higher score indicates a higher-quality translation.

According to Table 3, ARG performs better than CRF, which is the default choice for phrasing models. TEM is superior to both CRF and ARG. The next Sections examine whether ARG performance can be improved towards the TEM level, via a systematic method for defining new parameter values.

4.3 Optimization of ARG Parameters via PSO

After setting the weights manually to establish a baseline performance, PSO is tasked with determining the optimal attraction/repulsion weights. All weights take values that range from 0 to 1. Between the two swarm configurations (fully-informed and canonical), the fully-informed swarm achieved the best F-measure score, corresponding to a superior solution [16]. For this reason, in the present article only experiments with fully-informed swarms are reported, unless otherwise stated.

Table 4 summarises the results for the larger 80-particle swarm, by reporting the characteristics of the population formed by the 10 randomly-initialised runs. A substantial improvement is achieved over the manually-defined parameter set (cf. Table 2). The segmentation accuracy of each run of the PSO optimized system exceeded the best manual solution after fewer than 25 epochs. This demonstrates the fast convergence of PSO to a solution superior to the baseline. In addition, even the mean errors for each of the swarm configurations are substantially better that the best manually-tuned phrasing system, confirming the effectiveness of the PSO algorithm. The standard deviation of F-measure for the population of best solutions is very low over the 10 randomly-initialised runs, indicating that all runs consistently achieve solutions very close to the best solution.

To verify the effectiveness of the PSO optimization process, statistical tests are applied. The t-test is applied to determine whether the population of PSO-derived solutions is statistically better than the best manually-optimised solution. According to this test, the fully-informed swarm scores are significantly better than the manually-tuned one ($p < 0.0001$).

5 Previous Work on PSO Velocity Handling

Experimental results [16] using AdPSO have shown that exploration tails off quite early in experiments as the velocity vector is reduced to a relatively low magnitude, and most (if not all) particles gradually converge to a very limited area of the pattern space. When the swarm reaches this state, it is unable to discover radically new solutions. A likely solution is to reinitialise the velocity of the particles, to encourage exploration of the pattern space.

In the base PSO algorithm, parameters φ_1 and φ_2 are assigned fixed values to balance between exploration and exploitation. Later PSO variants have been aimed to improve the quality of solutions, this most frequently involving encouraging exploration versus exploitation. An inertia weight has been proposed [29] which is modified by a linear rule during the PSO optimisation process, so as to favour initially exploration and at the final stages turn towards exploitation. To ensure settling to a stable solution, Clerc and Kennedy [30] augment the velocity-update rule with a constriction factor that prevents the unbounded growth of particle velocity.

Garcia-Nieto et al. (2011) [31] introduce a velocity modulation algorithm, which controls the movement of the particles in each epoch so as not to exceed the limits of the problem domain. Evers et al. (2009) [32] aim to prevent premature convergence via a maximum swarm radius criterion. If a threshold is exceeded, then the swarm is regrouped at a radius around the best solution found so far, and the velocity component is also reinitialized. Helwig et al. [33] suggest recording the times a particle has succeeded in finding a new personal best, out of the last *n* epochs, to perform a more extensive search; otherwise the velocity is reduced to perform a narrower search. Chen and Li (2007) [34] introduce a stochastic velocity element to prevent the swarm from converging too quickly to a final solution. Finally Liu et al. [35] introduce a novel scheme that differentiates swarm particles into leaders (who perform exploration) and followers (who perform exploitation). A number of particles are teamed into a composite particle, with a velocity-anisotropic reflection (VAR) scheme used to stop velocities from reducing to very low values.

6 The Velocity Component in AdPSO

AdPSO [24] is chosen for further experimentation, since it is the more promising algorithm. The study of AdPSO simulations has revealed that as the swarm searches through the pattern space, gradually the magnitude of the velocity vector of individual particles decreases and converges to a value very close to 0.

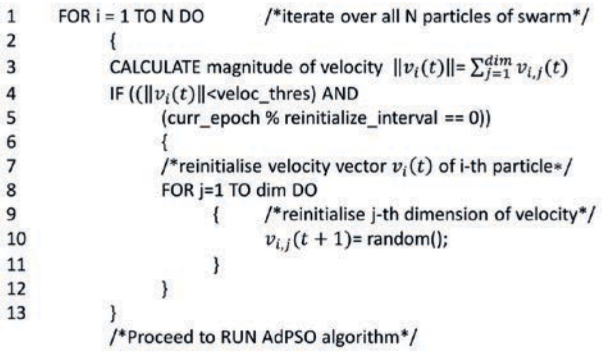
A typical run of the standard AdPSO algorithm is summarized by Figure 3, which presents for each epoch the average velocity over (1) all particles in the swarm, (2) the particles which have achieved a score equal to the global best and (3) the particles with a score inferior to the global best. The average velocity for all particles falls rapidly to low levels (less than 0.2, from a level of around 1.0), indicating that the swarm ceases to perform exploration and is restricted to exploitation. The average velocities of groups (2) and (3) are directly comparable in magnitude and all diminish to low levels. Hence, the stagnation phenomenon affects all particles in the swarm.

The proportion of swarm particles attaining a score equal to the global optimum for a given epoch is very high, far before the maximum number of iterations is reached. Concentration of particles to specific solutions becomes excessive, since in a 1000-epoch run from epoch 150 onwards, 50% of the particles converge to a solution equivalent to the global best, this percentage rising to 75% at epoch 550, indicating swarm stagnation.

To reinvigorate the search in the later phases, a scheme is defined to update the velocity periodically. Two aspects of the velocity component are studied, (i) reinitialising the velocity vector when a particle has stagnated and (ii) limiting the velocity magnitude during PSO operations so that the search is more gradual and stagnation is prevented.

6.1 Modification 1: Reinitialising the Velocity Component

The first potential improvement involves reinitialising the velocity vector when a given particle is found to have stagnated. Stagnation is assumed when the magnitude of velocity falls below a given threshold. This threshold, *veloc_thres*, is selectable by the users but is typically low (for a range of values k, the threshold is of the order of *0.1·k*). If the particle velocity drops below this threshold, the velocity is reinitialized.



```
1   FOR i = 1 TO N DO          /*iterate over all N particles of swarm*/
2       {
3       CALCULATE magnitude of velocity $\|v_i(t)\| = \sum_{j=1}^{dim} v_{i,j}(t)$
4       IF (($\|v_i(t)\|$<veloc_thres) AND
5           (curr_epoch % reinitialize_interval == 0))
6           {
7           /*reinitialise velocity vector $v_i(t)$ of i-th particle*/
8           FOR j=1 TO dim DO
9               {       /*reinitialise j-th dimension of velocity*/
10                      $v_{i,j}(t+1)$= random();
11              }
12          }
13      }
        /*Proceed to RUN AdPSO algorithm*/
```

Figure 4. Description of the velocity reinitialisation procedure

The velocity reinitialisation process is expressed using pseudocode in Figure 4. To ensure swarm stability, reinitialisation is only allowed at specific points during the PSO evolution (once every *a* epochs), and only after a number of *E* initial epochs have been carried out. Thus only at epochs *E+a*, *E+2a*, *E+3a*, etc. is it possible to reinitialize a stagnating particle's velocity component. The ability to to only reinitialise velocities evey *a* epochs is expressed by the modulo operator (%) used in step 5 of Figure 4.

6.2 Modification 2: Bounding the velocity vector to specified levels.

One observation from PSO test runs is that in several cases particles reach a state where along one or more dimensions, their parameter values are stuck to the minimum or maximum value very early on. This is due to a large initial velocity for the corresponding dimension, which sends the particle very rapidly to a specific region of the pattern space. Subsequently, the particle is unable to dislodge from this point due to the strong velocity component. A possible solution involves setting the magnitude $\|v(t)\|$ of velocity to a more moderate level, so that excessive movements in the pattern space in one step are discouraged and each particle uses a number of steps to transition from e.g. a very low value to a very high one along any given dimension.

7 Velocity Modulation Experiments

To reduce the simulation requirements, when evaluating the aforementioned modifications a relatively small AdPSO swarm was chosen, comprising 20 particles in total, allowed to evolve over a total of 1,000 epochs. For each swarm configuration, a set of 10 independent runs with different random initialisations was implemented. In the present article, the reinitialisation operation a is applied every 50, 25 or 10 epochs. The control result (baseline) is obtained by deactivating the velocity reinitalisation mechanism. The reinitialisation operator is only activated between epochs 300 and 900, to permit the swarm to settle to an initial solution during the first 300 iterations, and to then converge to a final solution in the last 100 iterations. Regarding the second improvement, the velocity magnitude $\|v(t)\|$ has been set to values of 0.125, 0.250, 0.333, 0.50, 1.00, 2.00 and 4.00. To obtain control results, simulations with an unrestricted velocity vector, without magnitude normalisation, were also run (denoted as "no-normal").

7.1 Evaluating the Velocity Reinitialisation Algorithm

The effectiveness of the PSO approach is reflected by the error attained by each swarm simulation. The statistics of the solution populations are expressed in Table 6, including the lowest and average error over 10 runs, when using no velocity reinitialisation (denoted as "no_reinit"), as compared to using a reinitialisation step every 50, 25 and 10 epochs. As can be seen, when using F-Measure the reinitialisation of velocity (second, third and fourth columns) contributes to lowering the error rate in comparison to using the standard AdPSO algorithm (1st column). The boxplots of Figure 5 also depict the characteristics of the solution populations when the F-Measure metric is used by AdPSO. The reduction of the phrasing error is evident when velocity reinitialisation is applied.

Similar results are recorded when using AdPSO with the Accuracy and F_2-Measure metrics (cf. Table 6, and also Figure 6 for the Accuracy metric). Thus when velocity reinitialisation is activated, both the average error as well as the lowest error are reduced for the metrics tested. One key difference is that for the Accuracy metric, the improvement achieved is reduced in comparison to that obtained with F-measure.

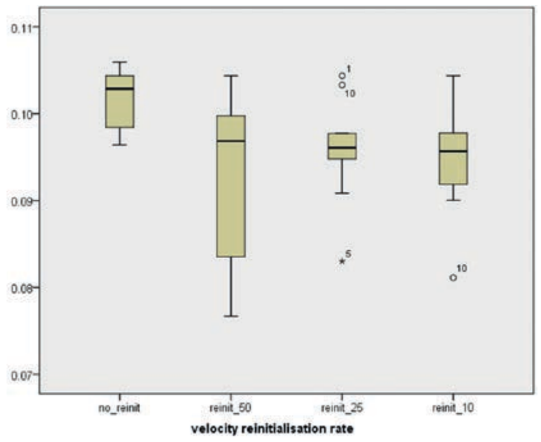


Figure 5. Boxplots of AdPSO phrasing errors using the F-measure metric. Velocity reinitialisation leads to a lower error than without reinitialisation ("no_reinit")

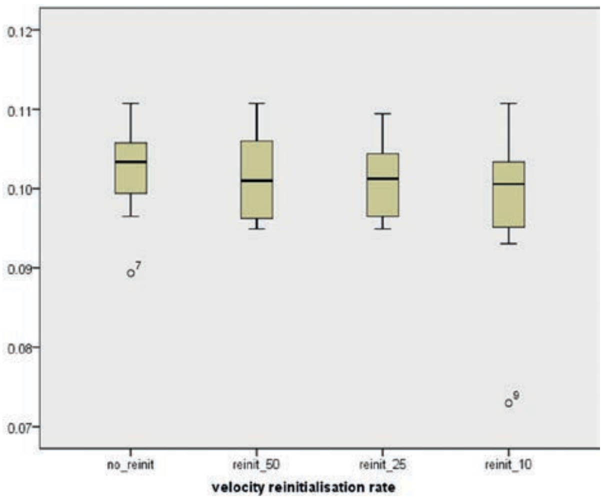


Figure 6. Boxplots of AdPSO best solutions in terms of error using the Accuracy metric

Paired-sample t-tests have been performed, to determine whether the addition of velocity reinitialisation contributes in a significantly improved AdPSO performance. When running AdPSO with F-measure, the error rate obtained with no reinitialisation (the baseline configuration) is significantly larger than with any of the three reinitialisation rates (cf. Table 5). With velocity reinitialisation, for rates of 50 and 25 the improvement over the baseline is significant at a 0.05 level of confidence. For a rate of 10, at a 0.01 level of confidence the accuracy is improved over the standard AdPSO algorithm (the baseline). On the contrary, the populations of solutions obtained with the three different reinitialisation rates are statistically equivalent to each other at a 0.05 level. When using the Accuracy metric, once again the reinitialisation leads to a numerical reduction of the error, though now the improvements over the baseline are statistically not significant (at a level of 0.05).

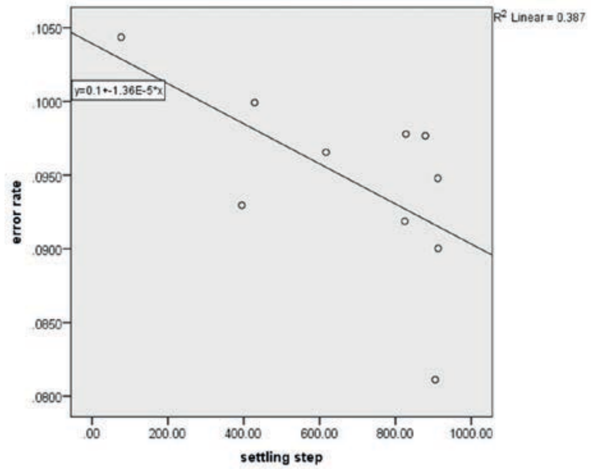


Figure 7. Relation between the swarm settling step and the final segmentation accuracy obtained, in the case of the F-Measure metric with reinitialisation every 10 epochs

As a second test, the correlation between the attained F-measure and the epoch at which the swarm reaches the best solution have been studied, using a linear regression test. Table 7 indicates in which epoch the optimal solution is achieved, when each of the three metrics is used with AdPSO. By combining Tables 6 and 7, a trend is observable of achieving lower error rates when more epochs of exploration are performed to reach the best solution. For instance, when using F-Measure to determine the error and without velocity reinitialisation, the average number of epochs to reach the best solution is 228 epochs, whilst for a reinitialisation rate of 50 a total of 508 epochs are needed. As expected, the activation of the reinitialisation algorithm results in the swarm utilizing more epochs to explore the search space to converge to a solution of a superior quality.

The actual correlation results for the 10 runs with the F-Measure metric are shown in Figure 7, for the reinitialisation rate of 0.10. A close to linear relation exists between the epoch where the minimal error is obtained and the error magnitude, showing that when more epochs are used effectively before convergence, a better optimization is achieved. Experimental results indicate that the velocity reinitialisation algorithm contributes to a longer (in terms of epochs) and more effective evolution of the swarm.

7.2 Evaluation of Velocity Normalisation

The second improvement concerns setting an upper bound to the initial velocity of each particle. For the present application, each weight takes values from 0.0 to 1.0 and thus the range along each dimension is equal to 1.0. Normalizing the magnitude of the velocity vector to a value lower than 1.0, ensures that the velocity along any dimension is substantially smaller than the range of values. This guides the swarm to settle more gradually. On the contrary, when the velocity magnitude is larger, in the first epoch (or the first few epochs) certain weights may take values at or beyond the maximum or minimum value of the range (1 or 0, respectively). These weights then are likely to remain fixed at this extreme value for the entire swarm evolution.

The levels of the velocity magnitude simulated are 0.125, 0.250, 0.333, 0.50, 1.00, 2.00 and 4.00. In addition, a configuration without velocity normalization has been run (this being depicted as "no_normal"). The results are summarized in Tables 8 and 9 for the three metrics. When using the F-Measure and Accuracy metrics, both the minimum error and the average error are substantially improved with small velocities (bounded at a 0.50 level), in comparison to an unconstrained velocity. The reduction in the error is substantial, amounting to 1.5%. On the contrary, when the velocity is bounded to a level higher than unity (for instance for a level of 4.00) then the optimized solution is inferior (the error rises). In the case of the F2-Measure, the improvements achieved for low velocity magnitudes are much smaller, though this is partly due to the lower error obtained for F2-Measure as compared to the other metrics.

The statistical characteristics of the solutions are depicted in more detail in Figures 8 and 9 respectively, when using the F-Measure and Accuracy metrics. Again, an improved performance (expressed as a reduced phrasing error) is obtained when the velocity magnitude is set to lower values.

A set of t-tests have been performed to determine whether this improvement in accuracy is significant. When using velocity normalization combined with the F-measure metric, the lower velocity magnitudes (namely those from 0.125 up to 0.50) give statistically equivalent results, with $p<0.01$. On the contrary, there is a statistically significant difference between the lower velocity magnitudes (5) and the higher magnitudes (of 2.0 or 4.0) at a level $p<0.001$. This indicates that lower velocities improve the effectiveness of the optimization process.

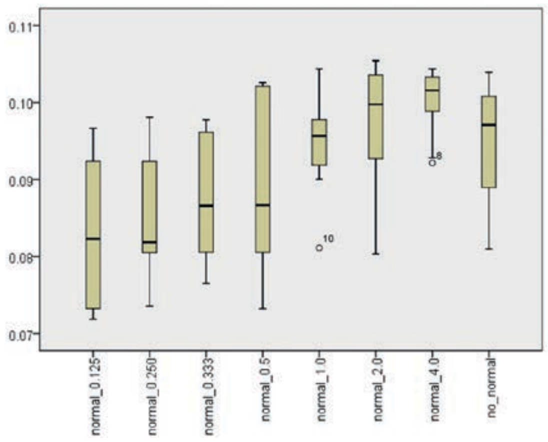


Figure 8. Boxplot of minimum error over 10 runs using the F-Measure metric for different normalisation levels of the velocity magnitude

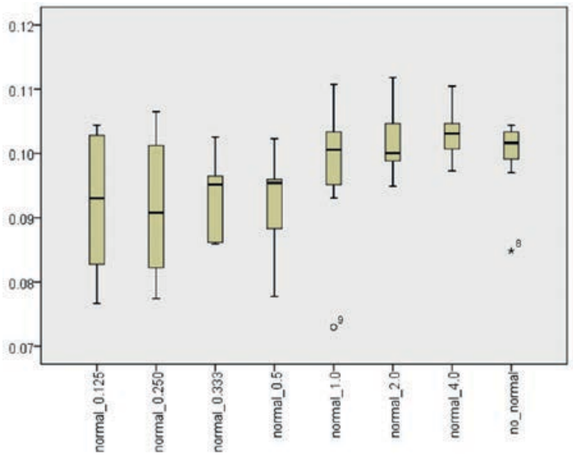


Figure 9. Boxplot of minimum error over 10 runs using the Accuracy metric for different normalisation levels of the velocity vector magnitude

Similarly, when using the Accuracy metric, a statistically significant improvement in error is achieved for lower velocity magnitudes ($\|v(t)\| \leq 0.5$), in comparison to higher values (equal to 2.0 or 4.0) or to using no bounds whatsoever. For instance, with a velocity magnitude of 0.125, a superior solution is obtained in comparison to a magnitude of 2.0 or 4.0 ($p<0.01$). For magnitudes of 0.250, 0.333 and 0.50, the population of solutions is superior to those for a magnitude of 2.0 ($p<0.02$) and to a magnitude of 4.0 ($p<0.001$). This analysis confirms that it is preferable to use a lower velocity magnitude (below 1.0), as this results in a statistically significant reduction in optimization error. The best results are obtained for normalisation levels of 0.125 and afterwards of 0.250, for all three metrics used.

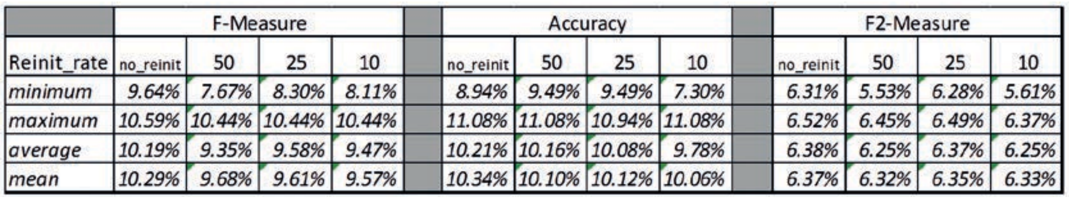
8 Conclusions

The present article has discussed the use of PSO-type algorithms to optimize the weights of a phrasing model based on attraction-repulsion principles. It has been found that PSO-type algorithms substantially improve the phrasing performance of a model based on attraction and repulsive forces. However, frequently the swarm converges prematurely to solutions, being unable to persevere the exploration of the pattern space to discover new solutions. As a result, new approaches to handle the velocity vector more appropriately have been designed and extensively tested.

Based on the results of these experiments, two main conclusions can be drawn. The first is that for splitting arbitrary sentences into syntactically-motivated phrases, the AdPSO performance can be improved. Improvements involve modifying the handling of the particles' velocity vectors. The first modification, which provides a major part of the improvement, concerns monitoring the velocity component of each particle and reinitializing it – if it falls below a certain level – to encourage the swarm's exploration effort. The second modification consists of bounding the magnitude of the velocity, so that the particle is prevented from finding a local-best solution very quickly, from which it is difficult to later disengage. Statistical analyses have shown that in many cases the proposed modifications provide statistically significant improvements in the solutions found.

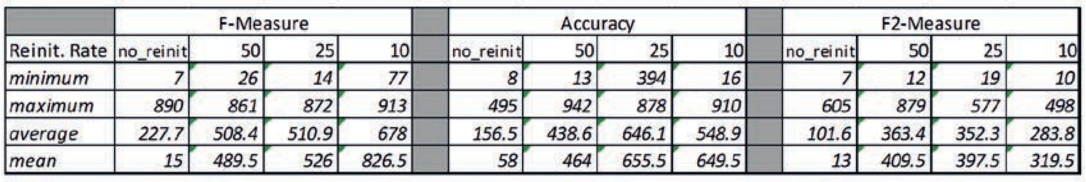
The experiments summarised in the present manuscript have focused on a single application. A future activity is to investigate whether similar im-

Table 6. Statistics of the phrasing error for each of the 3 metrics used, calculated over 10 independent runs



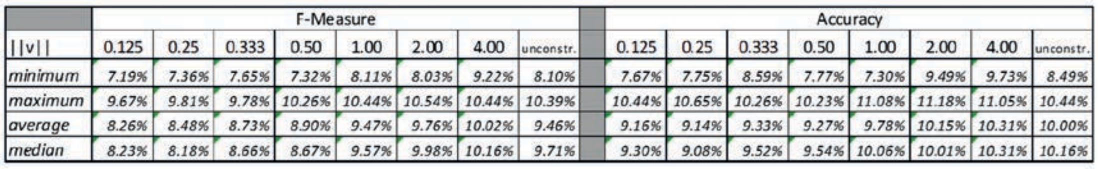
	F-Measure					Accuracy					F2-Measure			
Reinit_rate	no_reinit	50	25	10		no_reinit	50	25	10		no_reinit	50	25	10
minimum	*9.64%*	*7.67%*	*8.30%*	*8.11%*		*8.94%*	*9.49%*	*9.49%*	*7.30%*		*6.31%*	*5.53%*	*6.28%*	*5.61%*
maximum	*10.59%*	*10.44%*	*10.44%*	*10.44%*		*11.08%*	*11.08%*	*10.94%*	*11.08%*		*6.52%*	*6.45%*	*6.49%*	*6.37%*
average	*10.19%*	*9.35%*	*9.58%*	*9.47%*		*10.21%*	*10.16%*	*10.08%*	*9.78%*		*6.38%*	*6.25%*	*6.37%*	*6.25%*
mean	*10.29%*	*9.68%*	*9.61%*	*9.57%*		*10.34%*	*10.10%*	*10.12%*	*10.06%*		*6.37%*	*6.32%*	*6.35%*	*6.33%*

Table 7. Epoch where the best result is achieved for each of the 3 metrics used, calculated over 10 independent runs



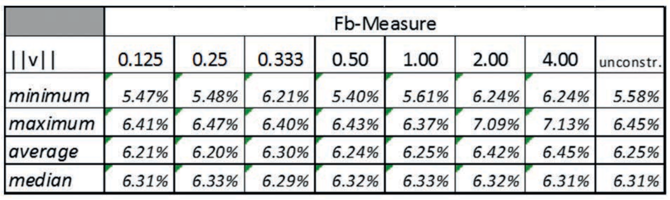
	F-Measure					Accuracy					F2-Measure			
Reinit. Rate	no_reinit	50	25	10		no_reinit	50	25	10		no_reinit	50	25	10
minimum	*7*	*26*	*14*	*77*		*8*	*13*	*394*	*16*		*7*	*12*	*19*	*10*
maximum	*890*	*861*	*872*	*913*		*495*	*942*	*878*	*910*		*605*	*879*	*577*	*498*
average	*227.7*	*508.4*	*510.9*	*678*		*156.5*	*438.6*	*646.1*	*548.9*		*101.6*	*363.4*	*352.3*	*283.8*
mean	*15*	*489.5*	*526*	*826.5*		*58*	*464*	*655.5*	*649.5*		*13*	*409.5*	*397.5*	*319.5*

Table 8. Effect of velocity magnitude on final AdPSO solutions for F-measure and Accuracy



	F-Measure									Accuracy							
||v||	0.125	0.25	0.333	0.50	1.00	2.00	4.00	unconstr.		0.125	0.25	0.333	0.50	1.00	2.00	4.00	unconstr.
minimum	*7.19%*	*7.36%*	*7.65%*	*7.32%*	*8.11%*	*8.03%*	*9.22%*	*8.10%*		*7.67%*	*7.75%*	*8.59%*	*7.77%*	*7.30%*	*9.49%*	*9.73%*	*8.49%*
maximum	*9.67%*	*9.81%*	*9.78%*	*10.26%*	*10.44%*	*10.54%*	*10.44%*	*10.39%*		*10.44%*	*10.65%*	*10.26%*	*10.23%*	*11.08%*	*11.18%*	*11.05%*	*10.44%*
average	*8.26%*	*8.48%*	*8.73%*	*8.90%*	*9.47%*	*9.76%*	*10.02%*	*9.46%*		*9.16%*	*9.14%*	*9.33%*	*9.27%*	*9.78%*	*10.15%*	*10.31%*	*10.00%*
median	*8.23%*	*8.18%*	*8.66%*	*8.67%*	*9.57%*	*9.98%*	*10.16%*	*9.71%*		*9.30%*	*9.08%*	*9.52%*	*9.54%*	*10.06%*	*10.01%*	*10.31%*	*10.16%*

Table 9. Effect of velocity magnitude on final AdPSO solutions for F_2_Measure



	Fb-Measure							
||v||	0.125	0.25	0.333	0.50	1.00	2.00	4.00	unconstr.
minimum	*5.47%*	*5.48%*	*6.21%*	*5.40%*	*5.61%*	*6.24%*	*6.24%*	*5.58%*
maximum	*6.41%*	*6.47%*	*6.40%*	*6.43%*	*6.37%*	*7.09%*	*7.13%*	*6.45%*
average	*6.21%*	*6.20%*	*6.30%*	*6.24%*	*6.25%*	*6.42%*	*6.45%*	*6.25%*
median	*6.31%*	*6.33%*	*6.29%*	*6.32%*	*6.33%*	*6.32%*	*6.31%*	*6.31%*

provements in performance can be achieved when AdPSO is applied to other real-world tasks. A second research activity would be to determine if similar improvements could be achieved by modifying the velocity component of other PSO variants.

References

[1] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Robert L. Mercer, The Mathematics of Statistical Machine Translation: Parameter Estimation. Computational Linguistics, Vol. 19, No.2, pp. 263-311, 1993.

[2] Philipp Koehn, Statistical Machine Translation. Cambridge University Press, Cambridge, 2010.

[3] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, Yoshua Bengio, On the properties of neural machine translation: Encoder–decoder approaches. Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 2014.

[4] George Tambouratzis, Sokratis Sofianopoulos Marina Vassiliou, Language-independent hybrid MT with PRESEMT. Proceedings of 2^{nd} HYTRA Workshop, held within the ACL-2013 Conference, Sofia, Bulgaria, pp. 123-130, 2013.

[5] James Kennedy, Russel C. Eberhart, Particle Swarm Optimization. Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, November, pp. 1942-1947, 1995.

[6] John Lafferty, Andrew McCallum, Fernando C.N. Pereira, Conditional Random Fields: Probabilistic Models for Segmenting and Labelling Sequence Data. Proceedings of ICML Conference, Williamstown, USA, pp. 282-289, 2001.

[7] Fei Sha, Fernando C.N. Pereira, Shallow Parsing with Conditional Random Fields. Proceedings of HLT-NAACL Conference, pp. 213-220, 2003.

[8] J.R. Finkel, A. Kleeman and C.D. Manning, Efficient, Feature-Based, Conditional Random Field Parsing. Proceedings of ACL-2008 Meeting, Columbus, Ohio, USA, pp. 959-967, 2008.

[9] Yoshimasa Tsuruoka, J. Tsujii, Sophia Ananiadou, Fast Full Parsing by Linear-Chain Conditional Random Fields. Proceedings of the 12th EACL Conference, Athens, Greece, pp. 790–798, 2009.

[10] Greg Durrett, Dan Klein, Neural CRF Parsing. Proceedings of the 53rd ACL Meeting, Beijing, China, pp. 302-312, 2015.

[11] George Tambouratzis, Conditional Random Fields versus template-matching in MT phrasing tasks involving sparse training data. Pattern Recognition Letters, Vol. 53, pp. 44-52, 2015.

[12] Elva J.H. Robinson, Francis L.W. Ratnieks, M. Holcombe, An agent-based model to investigate the roles of attractive and repellent pheromones in ant decision making during foraging. Journal of Theoretical Biology, Vol. 255, pp. 250-258, 2008.

[13] Changsheng Zhang, Jigui Sun, An Alternate two phases particle swarm optimization algorithm for flow shop scheduling problem. Expert Systems with Applications, Vol. 36, pp. 5162-5167, 2009.

[14] S.X. Yu, J. Shi, Segmentation with Pairwise Attraction and Repulsion. Proceedings of ICCV-2001 Conference, Vancouver, Canada, pp. 52–58, 2001.

[15] Karlheinz Stber, Petra Wagner, David Stall, Jens Helbig, Matthias Thomae, Jens Blauert, Wolfgang Hess, H. Mangold, Speech Synthesis by Multilevel Selection and Concatenation of Units from Large Speech Corpora. Verbmobil: Foundations of Speech-to-Speech Translation, Symbolic Computation, Springer, Berlin, pp. 519–537, 2000.

[16] George Tambouratzis, Applying PSO to Natural Language Processing Tasks: Optimizing the Identification of Syntactic Phrases. Proceedings of CEC-2016 Conference, Vancouver, Canada, pp. 1831-1838, 2016.

[17] Rui Mendes, James Kennedy, Jose Neves, The Fully Informed Particle Swarm: Simpler, Maybe Better. IEEE Transactions on Evolutionary Computation, 8 (3), 204-210, 2004.

[18] Moayed Daneshyari, Gary G. Yen, Cultural-Based Multiobjective Particle Swarm Optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol. 41, No. 2, pp.553-567, 2011.

[19] Xin Chen, Yangmin Li, A Modified PSO Structure Resulting in High Exploration Ability with Convergence Guaranteed. IEEE Transactions on Systems, Man & Cybernetics – Part B: Cybernetics, Vol. 37, No. 5, pp. 1271-1289, 2007.

[20] Jens Gimmler, Thomas Sttzle, Thomas E. Exner, Hybrid Particle Swarm Optimization: An examination of the influence of iterative improvement algorithms on its behavior. Proceedings of ANTS-2006 Workshop; Brussels, Belgium, 4-7 September, Lecture Notes in Computer Science, Vol. 4150, pp. 436-443, Springer, 2006.

[21] Richard P. Brent, Algorithms for Minimization without Derivatives, Englewood Cliffs, NJ, Prentice-Hall, Chapter 5, 1973.

[22] William H. Press, Saul A. Teukolsky, W.T. Vetterling, Brian P. Flannery, Numerical Recipes in Fortran 77: The Art of Scientific Computing. New York, U.S.A., Cambridge University Press, Chapter 7, 2002.

[23] James Kennedy, Rui Mendes, Neighborhood Topologies in Fully Informed and Best-of-Neighborhood Particle Swarms. IEEE Transactions on Systems, Man & Cybernetics – Part C: Applications and Reviews, Vol. 36, No. 4, pp. 515-519, 1996.

[24] Zhi-Hui Zhan, Jun Zhang, Yun Li, Henry Shu-Hung Chung, Adaptive Particle Swarm Optimization. IEEE Transactions on Systems, Man & Cybernetics – Part B: Cybernetics, Vol. 39, No.6, pp. 1362-1381, 2009.

[25] Helmut Schmid, Probabilistic part-of-speech tagging using decision trees. Proceedings of International Conference on New Methods in Language Processing, Manchester, UK. 1994.

[26] Helmut Schmid, Improvements in part-of speech tagging with an application to german. Proceedings of the ACL SIGDAT-Workshop. Dublin, Ireland, Association for Computing Machinery, 1995.

[27] Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu, BLEU: A Method for Automatic Evaluation of Machine Translation. 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, USA, pp. 311-318, 2002.

[28] NIST, 2002. Automatic Evaluation of Machine Translation Quality Using n-gram Co-occurrences Statistics.

[29] Yuhui Shi, Russel C. Eberhart, Parameter selection in particle swarm optimization, In: Proceedings of the 7th International Conference on Evolutionary Programming, New York, U.S.A., pp. 591-600, 1998.

[30] Maurice Clerc, James Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58-73, 2002.

[31] Jos Garca-Nieto, Enrique Alba, Restart particle swarm optimization with velocity modulation: a scalability test. Soft Computing, Vol.15, pp. 2221-2232, 2010.

[32] George I. Evers, Mounir Ben Ghalia, Regrouping particle swarm optimization: A new global optimization algorithm with improved performance consistency across benchmarks. IEEE International Conference on Systems, Man and Cybernetics, San Antonio, Texas, 11-14 October, pp. 3901-3908, 2009.

[33] Sabine Helwig, Frank Neumann, Rolf Wanka, Particle swarm optimization with velocity adaptation. International Conference on Adaptive and Intelligent Systems (ICAIS'09), 24 September, pp. 146-151, 2009.

[34] Xin Chen, Yangmin Li, A Modified PSO Structure Resulting in High Exploration Ability with Convergence Guaranteed. IEEE Transactions on Systems, Man & Cybernetics – Part B: Cybernetics, Vol. 37, No. 5, pp. 1271-1289, 2007.

[35] Lili Liu, Shengxiang Yang, Dingway Wang, Particle Swarm Optimization with Composite Particles in Dynamic Environments. IEEE Transactions on Systems, Man, And Cybernetics—Part B: Cybernetics, Vol. 40, No. 6, pp. 1634-1648, 2010.

George Tambouratzis received the Diploma in electrical engineering from the National Technical University of Athens Greece, in 1989, and the MSc degree in digital systems and the PhD degree in neural networks and pattern recognition, both from the Department of Electrical Engineering, Brunel University, U.K., in 1990 and 1993, respectively. Since 1996, he has been associated with the Institute for Language and Speech Processing, Athens, Greece, where he currently is a Senior Researcher. His research interests include neural networks, pattern recognition, evolutionary computation and computational linguistics with emphasis on machine translation. He is a member of the Technical Chamber of Greece and the IEEE.