EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

Research in Computing Science

Vol. 154 No. 3
March 2025

Research in Computing Science

***RESEARCH IN COMPUTING SCIENCE**, Año 25, Volumen 154, No. 3, Marzo de 2025, es una publicación mensual, editada por el Instituto Politécnico Nacional, a través del Centro de Investigación en Computación. Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738, Ciudad de México, Tel. 57 29 60 00, ext. 56571. https://www.rcs.cic.ipn.mx. Editor responsable: Dr. Grigori Sidorov. Reserva de Derechos al Uso Exclusivo del Título No. 04-2019-082310242100-203. ISSN: en trámite, otorgado por el Instituto Nacional del Derecho de Autor. Responsable de la última actualización de este número: el Centro de Investigación en Computación, Dr. Grigori Sidorov, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738. Fecha de última modificación 07 de Marzo de 2025.*

***RESEARCH IN COMPUTING SCIENCE**, Year 25, Volume 154, No. 3, March, 2025, is a monthly publication edited by the National Polytechnic Institute through the Center for Computing Research. Av. Juan de Dios Bátiz S/N, Esq. Miguel Othón de Mendizábal, Nueva Industrial Vallejo, C.P. 07738, Mexico City, Tel. 57 29 60 00, ext. 56571. https://www.rcs.cic.ipn.mx. Editor in charge: Dr. Grigori Sidorov. Reservation of Exclusive Use Rights of Title No. 04-2019-082310242100-203. ISSN: pending, granted by the National Copyright Institute. Responsible for the latest update of this issue: the Computer Research Center, Dr. Grigori Sidorov, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738. Last modified on March 7, 2025.*

Advances in Computing Science and Applications

Ana María Magdalena Saldaña Pérez
Carolina Palma-Preciado
Yanil Zuleima Contreras-Jiménez (eds.)

Instituto Politécnico Nacional, Centro de Investigación en Computación
México 2025

ISSN: in process

Copyright © Instituto Politécnico Nacional 2025
Formerly ISSNs: 1870-4069, 1665-9899

Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional "Adolfo López Mateos", Zacatenco
07738, México D.F., México

http://www.rcs.cic.ipn.mx
http://www.ipn.mx
http://www.cic.ipn.mx

Indexed in LATINDEX, DBLP and Periodica

Electronic edition

ISSN 1870-4069

Table of Contents

Preprocessing Method for Class Noise Treatment in Speech Emotion Recognition

ISSN 1870-4069

Randy Brandon Gallegos-Rodríguez[1],
Bernardo Garcia Bulle-Bueno[2],
Ana María Magdalena Saldaña-Pérez[1]

[1] Instituto Politécnico Nacional,
Centro de Investigación en Computación,
Mexico

[2] Institute of Data, Systems and Society,
MIT Cambridge,
USA

{rgallegosr2023, amagdasaldana}@cic.ipn.mx,
bernard0@mit.edu

Abstract. Speech emotion recognition datasets can have class noise due to subjectivity during the labeling process or because of automatic labeling. Class noise has been neglected until recently in machine learning research. In this work, we study the effects of controlled class noise in 10 datasets from different languages. We find that low levels of class noise (5-10%) do not significantly affect the performance of classifiers, but higher levels of class noise severely impact performance. Support Vector Machines (SVMs) appear to be the best candidate for handling class noise across most datasets and noise levels compared to other traditional algorithms. We propose a preprocessing method that effectively corrects mislabeled samples at moderate and high noise levels, enhancing the model's performance as measured by balanced accuracy.

Keywords: SER, class noise, support vector machines, preprocessing, ensemble.

1 Introduction

Speech Emotion Recognition (SER) is a task of affective computing, the subfield of artificial intelligence dedicated to recognizing human emotions, sentiments, and feelings [1]. The SER problem can be seen as a classification problem, where the aim is to find a proper model (classifier) that maps samples from the input space $\mathbf{X}$ to one of the c discrete labels or classes in a set of labels C. For the SER problem, as its name implies, the input samples are audio recordings, and the labels are the set of considered emotions. Fig. (1) is shows the most common methodologies used in SER. More details about the current state of the literature for the SER problem will be discussed in the *State of the Art* section.

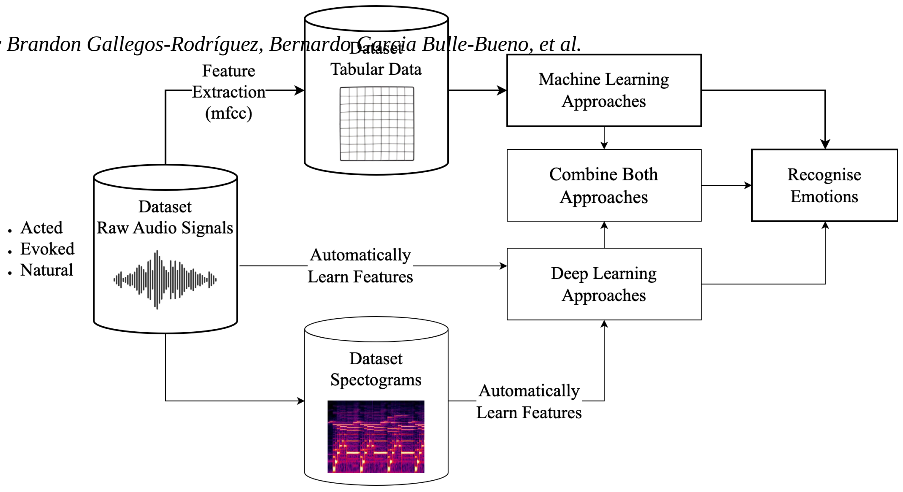


Fig. 1. Most common methodologies followed in the SER problem. Some strategies involve deep learning approaches. Thicker lines highlight the path followed in this work, consisting of extracting 13 Mel-frequency cepstral coefficients from audio recordings and implementing traditional machine learning algorithms. Image inspired by [2].

Since the SER problem can be treated as a classification problem, typical challenges in classification, such as noise, are also present in SER. Noise can be simply thought of as errors in the data; according to [8], it is also referred to as irregularities or corruptions in a dataset. For a more formal definition of noise, [9] briefly introduces the topic in the context of the Probably Approximately Correct (PAC) theory. PAC is a mathematical formalization of machine learning proposed by Valiant in [10]. For regression problems, the expected test error depends on variance, bias, and noise, as can be seen in Eq. 1:

$$Err(x_0) = \underbrace{[E\hat{f}(x_0) - f(x_0)]^2}_{Bias^2} + \underbrace{E[\hat{f}(x_0) - E\hat{f}(x_0)]^2}_{Variance} + \underbrace{\sigma^2}_{Noise}, \tag{1}$$

where x_0 is a sample from input space, $\hat{f}$ is the estimated function and f is the actual value according to the dataset. A complete derivation can be found in [11]. This result is studied in the bias-variance decomposition and is typically derived for regression due to its simplicity.

Noise can reduce the performance of the model, increase the complexity of the model, and extend the training time [8]. In [12], two categories of noise are presented: attribute noise and class noise. More details about these types of noise and methods for their treatment will be provided in the *State of the Art* section.

1.1 Outline of the Work

The remainder of this paper is structured as follows: In the next section, State of the Art, we provide a detailed review of the current literature on

SER and noise in classification tasks. Following this, the Research Goal section underscores the significance of addressing class noise in SER and outlines the objectives of this study. In Criteria for Effective Noise Treatment, we present a theoretical framework that guides the experimental design, which is detailed in the Experimental Framework section. The remainder of the paper follows the conventional structure of a scientific paper, including the presentation of results, conclusions and future work.

2 State of Art

2.1 Speech Emotion Recognition

In [2], a systematic literature review is presented. They consider seven key questions related to SER; one of these questions is, "What type of speech datasets[3] are used for SER?" They proposed three main categories: acted datasets, evoked or elicited datasets, and natural datasets.

Acted Datasets Acted datasets represent the most common type of SER datasets [2]. One example of an acted dataset is the Mexican Emotional Speech Database (MESD) [4]. For this dataset (and for many other acted datasets), a variety of actors (differing in age, gender, or accent) are chosen to speak words or phrases while acting out an emotion.

Elicited Datasets Elicited datasets are created by eliciting emotions in participants. One example of an elicited dataset is EmoMatchSpanishDB [5]. For this dataset, the labeling process was done through a crowdsourcing method.

Natural Datasets Natural datasets can be created in different ways, such as from TV shows, interviews, and conversations between customer care agents and customers [2, 7]. One example of this type of dataset is the RECOLA dataset, which is a multimodal corpus, meaning it includes more than one modality. In this case, it includes audio and video data, as well as physiological data, namely electrocardiogram and electrodermal activity. The data were obtained through a video conference where participants were asked to complete tasks.

Feature Extraction SER task follows the typical pipeline of any classification problem. If it is opted to extract features from the audio signal, instead of working directly with the raw signal, extracting Mel Frequency Cepstral Coefficients (MFCCs) is the most widely set of features employed [7]. According to [21], MFCCs are derived by capturing the envelope of the short-time power

[3] In the SER field, it is common to refer to the set of labeled audios as a database. In this work, we adopt the term *dataset*, more commonly used in other fields of machine learning.

spectrum, representing the vocal tract shape. The process involves segmenting utterances and converting them into the frequency domain using the short-time discrete Fourier transform to obtain MFCCs. Subsequently, Mel filter banks are employed to calculate energies in several sub-bands, followed by computing the logarithm of respective sub-band energies. Finally, MFCCs are obtained by applying the inverse Fourier transform.

Classification algorithms used in SER According to [3], there is currently no consensus on a state-of-the-art algorithm for SER that performs optimally under all conditions. It is recommended to conduct preliminary research to choose the most appropriate classification algorithm. For this reason, different algorithms will be tested in this work to compare their performance under specific noise conditions.

2.2 Noise in Classification Tasks

Attribute Noise Attribute noise refers to errors or corruptions in instances of the input space X; in particular, for the SER problem, this type of noise involves corruptions in the audio data. Attribute noise in SER has been extensively studied. For example, [13] examines the effect of white noise, [14] explores the impact of reverberation and Gaussian noise, and [15] presents a survey of SER in natural environments, which often include background noise.

In [16], a strategy is proposed to handle attribute noise for general machine learning tasks (tabular data), achieving new state-of-the-art results. Their strategy aims to correct attribute noise rather than deleting samples with potential noise. This approach is proposed because, as discussed in [12], removing noisy samples can eliminate noise but may also discard valuable information in some cases. Similar approaches, where noisy samples in unstructured data are corrected rather than removed, can be found in [14]. In this study, white noise in audio is handled with speech enhancement, followed by feature extraction to convert the problem to tabular classification.

Class Noise Class noise refers to errors or corruptions in the attribute class, i.e., having samples mislabelled. In the context of SER, this means having audio samples in the dataset labelled with an inappropriate emotion. [12] found that class noise is generally more harmful than attribute noise for classification tasks. This may be due to the fact that a mislabelled sample (x_i, y_i) completely meaningless, which is the information received by the algorithm during training.

In [17], SVMs were used to detect suspicious labels and correct them through expert supervision. This method showed improvement but may be unsustainable at a large scale. Other strategies involve eliminating noisy samples and are called filters. Examples of these filters include *edited nearest neighbor*, which removes samples inconsistent with their k nearest neighbors; filters based on the voting of ensemble methods; or filters that eliminate misclassified samples through cross-validation. For a more comprehensive review of filter methods, see [18].

According to [16], the two main strategies to handle both attribute and class noise are using robust algorithms (algorithms not sensitive to noise) or preprocessing techniques (such as filters).

3 Research Goal

3.1 Importance of Study Class Noise in SER Problem

In [5], it is mentioned that creating a SER dataset is expensive due to the need for actors. This limits the size of acted and elicited SER datasets. For this reason, deep learning approaches for SER have not yet reached their full potential [6]. These aspects highlight the need for new, scalable methods for creating SER datasets. For example, [19] exploits the current emergence of large-scale video available on social networks by implementing cross-modal techniques (audio and video in this case) and pseudo-labeling methods. In [6], data is augmented by using segments of labeled samples, but assigning the class of the utterance to the segment can create class noise. To address this problem, they use an iterative self-training method to label segments.

Another source of class noise in SER frequently discussed in the literature is the subjectivity of label annotators [20]. For all these reasons, class noise in SER problems has become an area of study. Current work addressing class noise in SER has improved performance by around 2% using BLSTM models [20]. Until recently, class noise in SER has been neglected, and there is still much work to be done. To our knowledge, there is no research on controlled class noise in SER across a significant number of datasets.

Objectives: This work has two primary objectives. The first is to study the robustness of several machine learning algorithms under controlled class noise conditions in the SER problem. We limit our study to well-known traditional algorithms, namely support vector machines (SVMs), random forest (RF), gradient boosting classifier (GB), AdaBoost, K-NN, and Ridge classifier. The second objective is to present a preprocessing method that satisfies the condition in Eq. (3) for specific noise levels. Finally, we compare the performance of the previously studied algorithms and our method alongside the most robust algorithm identified.

4 Criteria for Effective Noise Treatment

We propose the following formulation for the class noise problem, along with key conditions for a preprocessing method that aims to mitigate noise without discarding samples. For a single label classification dataset:

$$D = \{\mathbf{X}, \mathbf{Y}\} = \{\mathbf{x}_i, y_i\}_{i=0}^{N},$$

we say it's noise free if for every $\mathbf{x}_i \in \mathbf{X}$ its label y_i is equal to the ground-truth label for that sample, we can denote this dataset as:

$$D_0 = \{\mathbf{X}, \mathbf{Y}_0\} = \{\mathbf{x}_i, y_i^0\}_{i=0}^N.$$

Similarly, we can extent the notion of class noise in a dataset D_r by measuring it as the fraction r of mislabelled samples, r can take values from 0 to 1; for example, a dataset with ten percent of mislabelled samples could be express as $D_{0.3}$. By definition, it is true that for a given dataset D_r, Eq. (2) is true:

$$\text{Acc}(D_r) = \text{Acc}(\mathbf{Y}_r, \mathbf{Y}_0) = \frac{1}{N} \sum_{i=0}^{N} \delta\big(y_i^r == y_i^0\big) = 1 - r. \tag{2}$$

Note, that $\text{Acc}(D_r)$ presented in Eq. (2) is equivalent to accuracy, one well-known metric to measure performance in classification tasks. In this context, we are usign this measure (fraction of correct labels in a dataset) as a internal measure of class noise in a given dataset. For a real scenario, we can't calculate this measure since $\mathbf{Y}_0$ is unkonwn.

A preprocessing method $P(D_r)$ without removal of potential noisy samples, generates a new dataset $D_{\hat{r}} = (\mathbf{X}, \hat{\mathbf{Y}}_r)$ with the same samples, but some of them are re-labeled. We propose that P is a candidate to be a good preprocessing method for class noise treatment if it fulfills condition presented in Eq. (3), for some noise level r over a significant amount of datasets:

$$\text{Acc}\Bigg(P(D_r)\Bigg) \geq 1 - r, \tag{3}$$

nonetheless, this condition is only appropiate for balanced datasets, i.e., datasets where class distribution is highly skewed amog classes [30]. A dataset is said to be imbalanced if its imbalance ratio (IR), see Eq. (4), is smaller than 1.5 [31]:

$$IR = \frac{card\big(\text{majority class}\big)}{card\big(\text{minority class}\big)}. \tag{4}$$

For a more general purpose, another metric should be used, such as balanced accuracy, see Eq. (5), which is the average of sensitivity by classes, allowing an equal representation of all classes. For an imbalanced dataset, there can be a preprocessing method which fulfills the condition of Eq. (3) by misrepresenting minority classes in the process of re-labeling. For this reason, in this work we examine both metrics:

$$\text{BAcc}(D_r) = \frac{1}{|C|} \sum_{c \in C} \frac{1}{|\text{class c}|} \sum_{i=1}^{N} \delta\left(y_i^r == y_i^0 \wedge y_i^0 == c\right). \tag{5}$$

In Eq. (5) is the set of classes in D_r. We say, that $P(D_r)$ is a candiadte to be a good preprocessing method and not a actual one, since the final goal of noise treatment in classsification can be thought as improvement in model performance after applying P to the training set and testing in complete unseen data (test set), and not only improving mislabelled samples inside training set.

Table 1. Datasets used in this work. The reported cardinalities and imbalance ratios refer only to the samples utilized in this study and may differ from those reported in the original papers.

Dataset	Cardinality	Imbalance Ratio	Language	Type
Ravdess [22]	672	2.0	English	Acted
TESS [23]	1600	1.0	English	Acted
MESD [4]	574	1.0	Spanish	Acted
ShEMO [24]	2737	5.3	Persian	Semi-Natural
CaFE [25]	504	2.0	Canadian-French	Acted
EMO-DB [26]	339	2.0	German	Acted
CREMA-D [27]	4900	1.2	English	Acted
EMOVO [28]	336	1.0	Italian	Acted
EmoMatchSpanishDB [5]	1318	1.8	Spanish	Elicited
URDU [29]	400	1.0	Urdu	Natural

5 Experimental Framework

5.1 Algorithms' Robustness

The data for this study comprised 10 supervised SER datasets, as shown in Tab. 1. We only considered the emotions common to all datasets: anger, happiness, neutral, and sadness. All audio files were downsampled to a frequency of 16 kHz and converted to mono-channel signals. After standardizing the audio signals, 13 Mel-frequency cepstral coefficients (MFCCs) were extracted from each, transforming the unstructured data into tabular form. Various levels of class noise were introduced: 0%, 5%, 10%, 20%, 30%, 40%, and 50%. Noise was injected randomly, ensuring each class contained approximately the same amount. This resulted in a total of 60 datasets, with 10 datasets per noise level. This approach allowed us to control the noise levels in datasets assumed to be correctly labeled, enabling an analysis of the effects of varying noise levels on different algorithms.

To ensure a fair comparison between algorithms, we tested various hyperparameter instances for each classification scenario (SER dataset and noise level). A grid search was used to find the optimal set of hyperparameters. Despite using a substantial number of SER datasets, we performed 3×5-fold cross-validation. Within each 5-fold cross-validation, different noise injections, folds, and random states for algorithms were employed. Balanced accuracy was chosen as the evaluation metric, given that half of the datasets are imbalanced (see Tab. 1). The grid search space was selected based on prior knowledge of key hyperparameters for regularization in each algorithm.

5.2 Preprocessing Method Proposed

After evaluating the robustness of the algorithms, we found that SVM often performed the best, particularly with a radial basis function (RBF) kernel and a regularization parameter $C = 10$. The second-best algorithms were RF and

Table 2. Overall results of the preprocessing method's impact are summarized as follows. Datasets Improved show the number of cases (out of 10) with improved correct labels. The overall change represents the average change across all datasets for each noise level.

Level of Noise	0	5	10	20	30	40	50	(%)
Datasets Improved	0	5	6	7	9	9	10	
Overall Change	-4.6	-2.2	-0.1	4.0	10.0	13.2	17.8	(%)

k-NN, which showed similar performance across all noise levels. However, k-NN can be implemented using KD-trees, making it significantly faster than RF. Therefore, we decided to use both algorithms in an ensemble approach to detect noisy samples and re-label them. The process is as follows:

Given a noisy dataset D_r with an unknown noise level r and unknown ground-truth labels $\mathbf{Y}0$ (which are not available in real scenarios), two models, $hsvm$ and h_{knn}, are trained on $(\mathbf{X}, \mathbf{Y}_r)$. These models are then used to estimate the emotion for each sample across D_r, re-labeling them as indicated in Eq. (6):

$$P(\mathbf{x}_i) = \begin{cases} h_{svm}(x_i) & h_{svm}(x_i) = h_{knn}(x_i) \\ y_i^r & \text{otherwise} \end{cases} . \tag{6}$$

To test whether the proposed method is a valid class noise corrector, we applied it to the 10 datasets under all noise conditions. We assessed whether the similarity, based on 2 and 5, between $\hat{\mathbf{Y}}_r$ and $\mathbf{Y}_0$ improved compared to $\mathbf{Y}_r$ and $\mathbf{Y}_0$.

To evaluate whether the proposed method results in actual improvements in model performance, rather than merely correcting a fraction of mislabeled data, we tested the method within the same cross-validation framework used to assess algorithm robustness. Specifically, for each dataset D_r, 5-fold cross-validation was employed. In each iteration, the preprocessing method P was applied to the four training folds. An SVC was then trained on the re-labeled training folds and used to estimate emotions in the testing fold. Performance was measured using the unknown $\mathbf{Y}_0$. This process was repeated three times for each dataset and noise level, each time with different random injections and folds.

6 Results and Discussion

6.1 Algorithms Robustness

Results for the best hyperparameter configurations tested for each algorithm are shown in Fig. (5), along with the results of the preprocessing method. We found that the robustness of some algorithms depends significantly on the chosen hyperparameters. Fig. (2) displays the performance for two instances of k-NN and SVMs.

As shown in Fig. (2), k-NN's performance does not vary significantly with the number of neighbors, k. Although a greater number of neighbors is required as

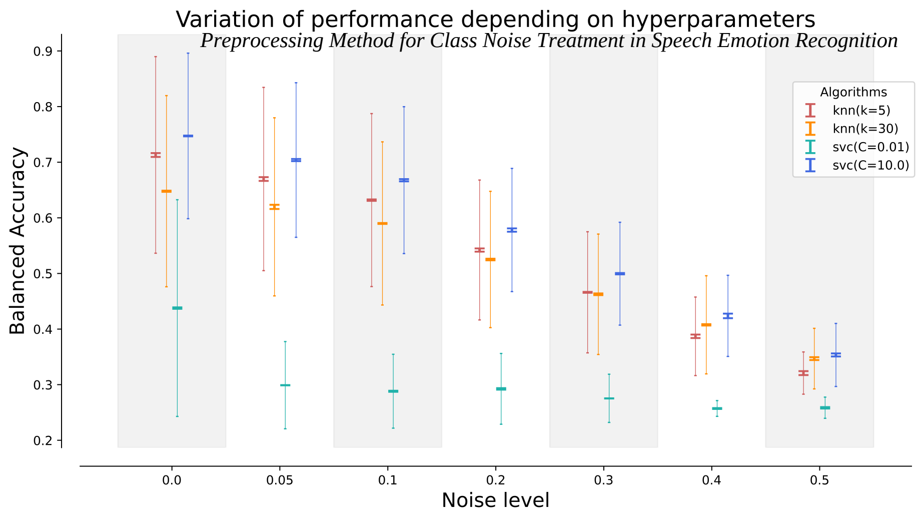


Fig. 2. Performances for each algorithm and different evaluations at one hyperparameter are displayed with double error bars. Thicker, smaller error bars represent the standard deviation of the mean performance across all datasets for each 5-fold cross-validation. Larger error bars represent the standard deviation across all datasets.

noise increases, the performance changes are less pronounced compared to SVMs with respect to the regularization parameter C. For each model, we selected the best hyperparameter configuration across all datasets and noise levels. Generally, SVM was the best algorithm for all noise levels, particularly with balanced class weights, an RBF kernel, and $C = 10$. For some datasets, k-NN and random forest performed best at low noise levels. Our results are publicly available on GitHub for more detailed analysis.

6.2 Impact of Preprocessing Method at Different Levels of Noise

To measure the effects of the preprocessing method under noisy conditions, we estimate the changes in the percentage of correct labels, as defined in Eq. (2). Fig. (3) shows the changes in extreme cases: no noise and 50% noise injection.

It can be seen that the preprocessing method improves the labeling of samples with 50% noise across all datasets but mislabels samples when applied in noise-free conditions. A summary of the results is provided in Tab. (2).

From Tab. (2), it is evident that the proposed method shows improvements for conditions with at least 20% noise injection. The enhancements in correct labels are more significant under high noise conditions compared to the decreases observed in low noise conditions.

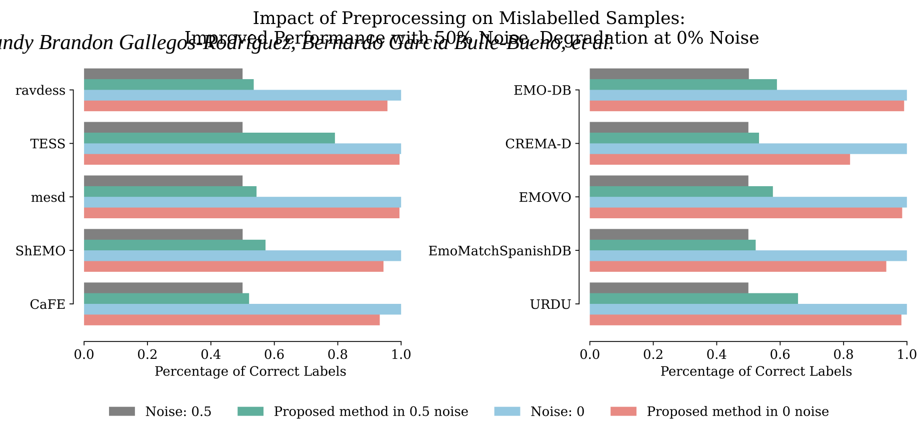


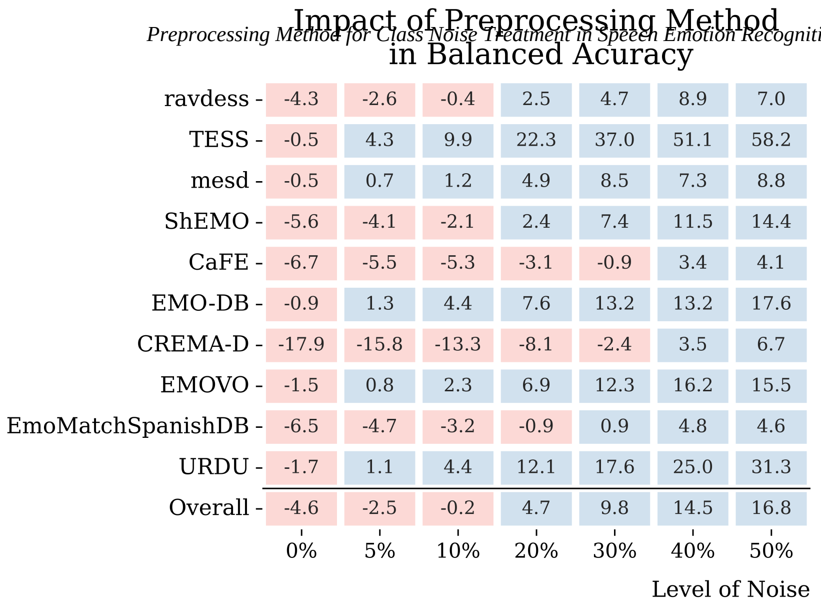
Fig. 3. The percentage of correct labels shown is the average from repeating the process three times for each dataset and noise level.

To assess whether these improvements compromise minority classes, we compared the percentage changes between $\mathrm{BAcc}(\mathbf{Y}_r, \mathbf{Y}_0)$ and $\mathrm{BAcc}(\hat{\mathbf{Y}}_r, \mathbf{Y}_0)$. The results, shown in Fig. (4), are consistent with those presented in Tab. (2).

According to Fig. (4), the proposed method may not be suitable for low noise levels (0-10%) due to a decrease in similarity to the ground-truth labels. For moderate noise levels (20-30%), the method appears to be effective, with most datasets showing improvement. Notably, the ShEMO dataset, which is the most imbalanced with a ratio of 5.3, also shows improvement at these noise levels, suggesting that the method may be resilient to imbalanced data. Finally, for higher noise levels (40-50%), the proposed method delivers better and more consistent results across all datasets.

6.3 Improvement in Model's Performance on Cross Validation Settings

After confirming that the proposed method is a viable preprocessing option for certain noise levels according to our definition, we need to ensure that these changes result in actual improvements in model performance. We followed the procedure described in the experimental framework. Results are shown in Fig. (5), along with the most robust configurations for each algorithm studied. Fig. (5) indicates that the proposed method performs worse under noise-free conditions, falling below even SVMs and most other algorithms. At 5% noise, the method remains nearly resilient and shows performance close to that of SVMs. At 10% noise, the proposed method achieves the best results compared to all other algorithms. For higher noise levels, while the performance of the preprocessing technique begins to degrade, it remains significantly better than that of all other algorithms.



	0%	5%	10%	20%	30%	40%	50%
ravdess	-4.3	-2.6	-0.4	2.5	4.7	8.9	7.0
TESS	-0.5	4.3	9.9	22.3	37.0	51.1	58.2
mesd	-0.5	0.7	1.2	4.9	8.5	7.3	8.8
ShEMO	-5.6	-4.1	-2.1	2.4	7.4	11.5	14.4
CaFE	-6.7	-5.5	-5.3	-3.1	-0.9	3.4	4.1
EMO-DB	-0.9	1.3	4.4	7.6	13.2	13.2	17.6
CREMA-D	-17.9	-15.8	-13.3	-8.1	-2.4	3.5	6.7
EMOVO	-1.5	0.8	2.3	6.9	12.3	16.2	15.5
EmoMatchSpanishDB	-6.5	-4.7	-3.2	-0.9	0.9	4.8	4.6
URDU	-1.7	1.1	4.4	12.1	17.6	25.0	31.3
Overall	-4.6	-2.5	-0.2	4.7	9.8	14.5	16.8

Fig. 4. Percentage changes in balanced accuracy serve as an internal measure for dataset D_r, estimating the similarity between $\mathbf{Y}_r$ and $\mathbf{Y}_0$ before applying the preprocessing method and $\hat{\mathbf{Y}}_r$ and $\mathbf{Y}_0$ after applying the method.

7 Conclusions and Future Work

In this work, we conducted a systematic study on the effects of noise in the Speech Emotion Recognition (SER) problem. Our primary contributions are twofold. First, we established that among the algorithms studied, Support Vector Machines (SVM) consistently deliver superior performance across both low and high noise levels. Second, we proposed a preprocessing technique that effectively outperforms traditional machine learning algorithms in high noise conditions.

The proposed method has shown promise in enhancing data quality for SER datasets. It significantly improves performance in noisy environments, outperforming all other algorithms studied, showing its potential as a valuable tool for preprocessing in real-world applications.

Future work will focus on enhancing the generalizability of this method for cross-corpora problems, where models are trained and tested on datasets with different distributions, such as actors or speakers from various regions or languages. We aim to investigate how well this preprocessing technique can support a broader range of algorithms, including neural networks, which were not covered in this study. By extending the scope of our research, we hope to improve the robustness and adaptability of SER systems across diverse and challenging conditions.

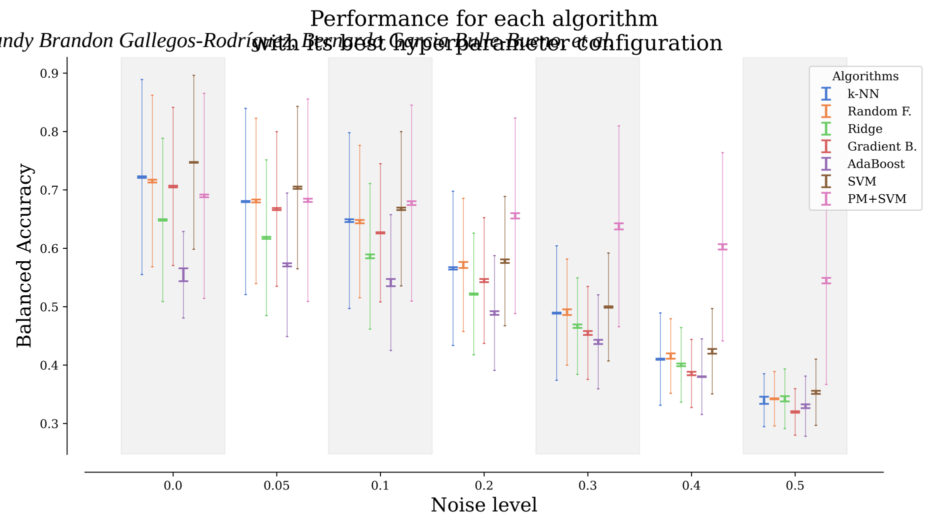


Fig. 5. Performance results for each algorithm are displayed with double error bars. Thicker, smaller error bars represent the standard deviation of the mean performance over all datasets for each 5-fold cross-validation. Larger error bars represent the standard deviation across all datasets. The proposed method is labeled as 'PM'.

References

1. Afzal, S., Khan, H. A., Piran, M. J., Lee, J. W.: A comprehensive survey on affective computing; challenges, trends, applications, and future directions. IEEE Access (2024)
2. Singh, Y. B., Goel, S.: A systematic literature review of speech emotion recognition approaches. Neurocomputing, 492, 245–263 (2022)
3. Hashem, A., Arif, M., Alghamdi, M.: Speech emotion recognition approaches: A systematic review. Speech Communication, 102974 (2023)
4. Duville, M. M., Alonso-Valerdi, L. M., Ibarra-Zarate, D. I.: The Mexican Emotional Speech Database (MESD): elaboration and assessment based on machine learning. In: 2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC), pp. 1644–1647, IEEE (2021)
5. Garcia-Cuesta, E., Salvador, A. B., Pãez, D. G.: EmoMatchSpanishDB: study of speech emotion recognition machine learning models in a new Spanish elicited database. Multimedia Tools and Applications, 83(5), 13093–13112 (2024)
6. Mao, S., Ching, P. C., Lee, T.: Enhancing segment-based speech emotion recognition by iterative self-learning. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 30, 123–134 (2021)
7. Wani, T. M., Gunawan, T. S., Qadri, S. A. A., Kartiwi, M., Ambikairajah, E.: A comprehensive review of speech emotion recognition systems. IEEE access, 9, 47795–47814 (2021)
8. Hasan, R., Chu, C.: Noise in Datasets: What Are the Impacts on Classification Performance?[Noise in Datasets: What Are the Impacts on Classification Performance? In: Proceedings of the 11th International Conference on Pattern Recognition Applications and Methods (2021)

9. Mohri, M., et al.: Foundations of Machine Learning. Cambridge, Massachusetts, The Mit Press (2018)
10. Valiant, L. G.: A theory of the learnable. Communications of the ACM, 27(11), 1134–1142 (1984)
11. Hastie, T., Tibshirani, R., Friedman, J. H., Friedman, J. H.: The elements of statistical learning: data mining, inference, and prediction, Vol. 2, pp. 1–758, New York: Springer (2009)
12. Zhu, X., Wu, X.: Class noise vs. attribute noise: A quantitative study. Artificial intelligence review, 22, 177–210 (2009)
13. Huang, C., Chen, G., Yu, H., Bao, Y, Zhao, L.: Speech emotion recognition under white noise. Archives of Acoustics, 38(4), pp. 457–463 (2017)
14. Heracleous, P., Yasuda, K., Sugaya, F., Yoneyama, A., Hashimoto, M.: Speech emotion recognition in noisy and reverberant environments. In 2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII), pp. 262-266, IEEE (2017)
15. Fahad, M. S., Ranjan, A., Yadav, J., Deepak, A.: A survey of speech emotion recognition in natural environment. Digital signal processing, 110, 102951 (2021)
16. Sáez, J. A., Corchado, E.: ANCES: A novel method to repair attribute noise in classification problems. Pattern Recognition, 121, 108198 (2022)
17. Ekambaram, R., Fefilatyev, S., Shreve, M., Kramer, K., Hall, L. O., Goldgof, D. B., Kasturi, R.: Active cleaning of label noise. Pattern Recognition, 51, 463–480 (2016)
18. Chen, Q., Jiang, G., Cao, F., Men, C., Wang, W.: A general elevating framework for label noise filters. Pattern Recognition, 147, 110072 (2024)
19. Zhang, S., Chen, M., Chen, J., Li, Y. F., Wu, Y., Li, M., Zhu, C.: Combining cross-modal knowledge transfer and semi-supervised learning for speech emotion recognition. Knowledge-Based Systems, 229, 107340 (2021)
20. Fujioka, T., Homma, T., Nagamatsu, K.: Meta-learning for speech emotion recognition considering ambiguity of emotion labels. In: INTERSPEECH (2020)
21. Gupta, D., Bansal, P., Choudhary, K.: The state of the art of feature extraction techniques in speech recognition. Speech and Language Processing for Human-Machine Communications: Proceedings of CSI 2015, 195-207 (2018)
22. Livingstone, S.R., Russo, F.A.: The Ryerson Audio–Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5):e0196391 (2018)
23. Dupuis, K., Pichora-Fuller, M.: Recognition of emotional speech for younger and older talkers: Behavioural findings from the Toronto emotional speech set. Canadian Acoustics, 39(3):182-3. Available from: https://jcaa.caa-aca.ca/index.php/jcaa/article/view/2471 (2011)
24. Mohamad Nezami, O., Jamshid Lou, P., Karami, M.: ShEMO: A large-scale validated database for Persian speech emotion detection. Language Resources and Evaluation, 53, 1–16 (2019)
25. Gournay, P., Lahaie, O., Lefebvre, R.: A Canadian French emotional speech dataset. In: Proceedings of the 9th ACM multimedia systems conference, pp. 399–402 (2018)
26. Burkhardt, F., Paeschke, A., Rolfes, M., Sendlmeier, W. F., Weiss, B.: A database of German emotional speech. In: Interspeech, Vol. 5, pp. 1517–1520 (2005)
27. Cao, H., Cooper, D. G., Keutmann, M. K., Gur, R. C., Nenkova, A., Verma, R.: Crema-d: Crowd-sourced emotional multimodal actors dataset. IEEE transactions on affective computing, 5(4), 377–390 (2014)

28. Costantini, G., Iaderola, I., Paoloni, A., Todisco, M.: EMOVO corpus: An Italian emotional speech database. In: Proceedings of the ninth international conference on language resources and evaluation (LREC'14), pp. 3501–3504, European Language Resources Association (ELRA) (2014)
29. Latif, S., Qayyum, A., Usman, M., Qadir, J.: Cross lingual speech emotion recognition: Urdu vs. Western languages. In: 2018 International conference on frontiers of information technology (FIT), pp. 88–93, IEEE (2018)
30. Alcal-Fdez, J., Fernndez, A., Luengo, J., Derrac, J., Garca, S., Snchez, L., Herrera, F.: Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing, 17(2-3), 255–287 (2011)
31. Villuendas-Rey, Y., Yáñez-Márquez, C., Camacho-Nieto, O.: Ant-based feature and instance selection for multiclass imbalanced data. IEEE Access (2024)

ISSN 1870-4069

Particle-based Simulations of Liquid Crystals Supported by GPU Parallelization in CUDA

Jorge Fierro[1], Humberto Híjar[1,2]

[1] Universidad La Salle México, Centro de Investigación, Ciudad de México, Mexico

[2] Universidad Nacional Autónoma de México, Facultad de Ciencias, Investigación Científica, Ciudad de México, Mexico

jorge.fierro@lasalle.mx, humberto.hijar@lasalle.mx

Abstract. Liquid crystals are fluids that show certain amount of order in the orientation and position of their molecules in contrast with simple fluids where both types of order are absent. They have been subject of numerous studies due to their technological relevance. In this research work, it is proposed a method for simulating the liquid crystal phase with the simplest symmetry, known as the nematic phase. The method is based on particles that interact in independent sets, which allows to propose programming it in parallel. This is done in Graphic Processing Units (GPUs) on NVIDIA's CUDA architecture. It is shown that the method allows to simulate the appearance of molecular order on reproducible conditions. It is also clearly exhibited that the parallel procedure has a much higher performance than that given by the serial version of the same simulation algorithm.

Keywords: Liquid crystal, simulation, GPU parallelization.

1 Introduction

Introductory physics describes three states of matter: solid, liquid, and gas, which are differentiated by the amount of order shown by their molecules [5]. Usually, materials transition between the solid and liquid states without an intermediate stage. However, in the late 19th century, Freiderich Reinitzer discovered that intermediate phases can exist between these two states [4]. A few years later, Otto Lehmann named these phases as we know them today: liquid crystals [4].

The applications of liquid crystals include displays for televisions, cell phones, and computers known as LCDs (Liquid Crystal Displays); tunable wavelength filters; resonant cavities for tunable lasers; thermometers, and smart windows [15]. Recent research suggests their use in detecting pathogens, antigens, cancerous tumors [21, 22], as well as in controlling the trajectory of microorganisms [16].

A vast variety of liquid crystals is known. All of them are formed by molecules whose symmetry is not spherical, e.g., elongated rod-shaped molecules as those of N–(4–Methoxybenzylidene)–4–butylaniline (commonly referred to as MBBA) illustrated in Figure 1 (a), or 4–Cyano–4′–pentylbiphenyl (customarily known as 5CB) illustrated in Figure 1 (b). When atomistic details are not relevant, these molecules can be modeled as rigid rods as those illustrated in Figure 2.

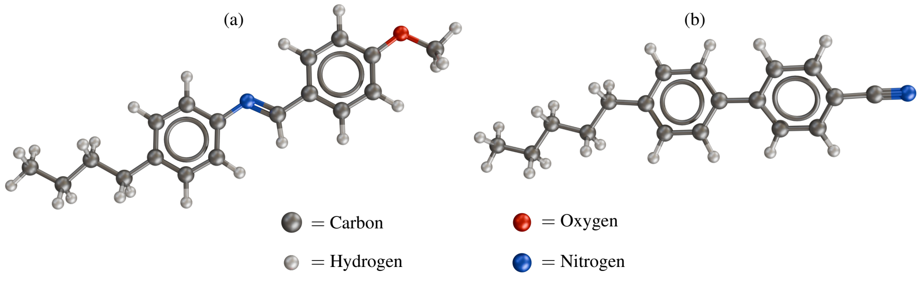


Fig. 1. Two elongated molecules that form liquid crystal phases: MBBA (a) and 5CB (b). Their constituents atoms can be identified by the levels at the bottom. The central part of both molecules contains benzene rings (flat hexagons with six carbon atoms) that give them a rather rigid structure.

The liquid crystal phase with the simplest symmetry is known as the nematic phase. The molecular arrangement in a nematic liquid crystal (NLC) is schematically illustrated in Figure 2, where it is compared against that of the crystal and isotropic liquid phases. In a crystal, molecules are perfectly positioned at the nodes a periodic lattice and all of them point along the same direction. In the completely opposite case, corresponding the a simple or isotropic liquid, molecules move arbitrarily through the sample and they point in every direction with the same probability. In a NLC, the centers of mass of the molecules move arbitrarily, as in a simple liquid, but the molecular axes remain oriented around a common axis known as the *director*, represented by a unit vector, $\hat{\mathbf{n}}$, [11]. Thus, NLCs are states of matter with an intermediate order between that of crystal a that of usual liquid. The reader is referred to reference [9] (in Spanish) where concepts and mathematical aspects of liquid crystal phases are discussed in an introductory manner.

For the previously mentioned reasons, liquid crystals are of great interest in applied sciences and materials engineering, where computational simulations have played a crucial role in understanding their properties due to their ability to test a wide range of system's parameters and to handle conditions that are hard to achieve experimentally [1]. Recently, an algorithm known as Nematic Multiparticle Collision Dynamics (N-MPCD) has been proposed, which describes the NLC as a system of particles that carry an orientation vector [19]. Periodically, the particles are allowed to interact with those in their vicinity. To do this, the simulation space is subdivided into contiguous cubic cells within which independent operations are performed, suggesting that the method could be parallelized. Two alternative variants of N-MPCD are known. One is due to Shendruk and

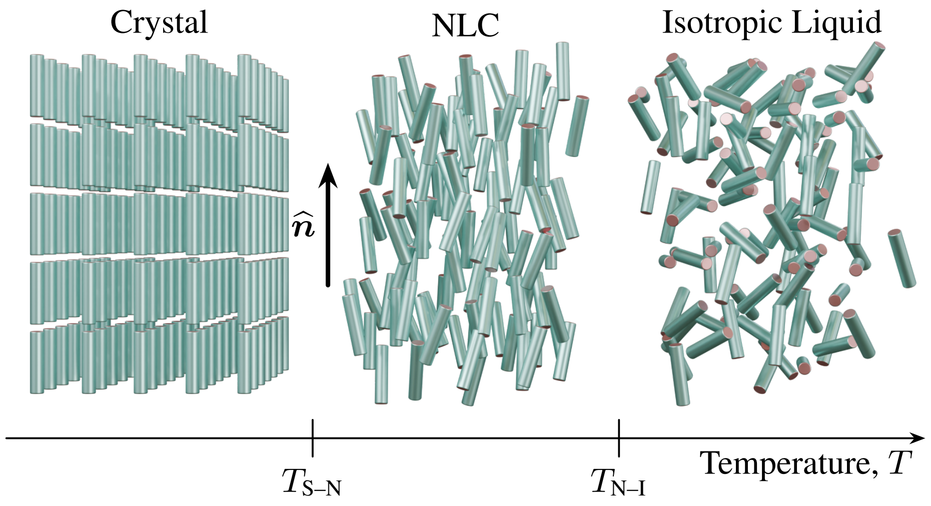


Fig. 2. Schematic of the molecular order in three phases of matter: crystal (left), NLC (center), and isotropic liquid (right). These phase occur as function of temperature where $T_{\mathrm{S-N}}$ and $T_{\mathrm{N-I}}$ indicate transitions from solid to nematic and from nematic to isotropic liquid, respectively. Vector $\hat{\mathbf{n}}$ represents the average molecular orientation in the liquid crystal phase.

Yeomans [19] and is based on a collision operator that promotes reorientation of particles dictated by a local mean-field potential to achieve nematic order. The other one, due to the Mazza and coworkers [14], simulates nematic order by incorporating explicit hydrodynamic equations of liquid crystals.

N-MPCD is an extension of a simple fluid simulation method known as Multi-particle Collision Dynamics (MPCD), for which various algorithms that operate in parallel have been proposed. One of the first was developed by Petersen et al., who adapted the method to run on multiple processors of a Cray XT3 computer [17]. Westphal et al. have developed an MPCD implementation based on graphics processing units (GPUs), achieving a performance gain of up to two orders of magnitude compared to a comparable version on central processing units (CPUs) [23]. Howard et al. have presented an open-source implementation of the algorithm that scales to run on hundreds of GPUs [13, 12]. Halver et al. have used heterogeneous GPU nodes to parallelize MPCD in an implementation based on Cabana [7]. More recently, Ratan has created parallelized simulations based on an hybrid Molecular Dynamics–MPCD scheme to investigate the behavior of active matter systems [18].

The aim of the present work is to parallelize the N-MPCD method, in the Shendruk and Yeomans version, taking advantage of the fact that it works with quantities representing particles grouped in discrete and independent spaces. The problem to be solved consists of performing operations on the properties of the particles that make up the system simultaneously and independently when necessary, as well as performing parallel convergent operations that require particles grouped in the system's cubic cells. All of this must be done while respecting the physical and mathematical rules that produce nematic behavior in the system. The goal is for this simulation to run on GPUs. Additionally, the performance of this parallel implementation is expected to surpass that of a previously developed serial version [8, 10].

One of the main challenges in developing the GPU-parallelized N-MPCD method was that many processing threads needed to write to the same memory section simultaneously. This problem was solved by using a processing thread for each collision cell, dedicated exclusively to averaging the properties of the particles contained within it. This implementation improved computation time by an order of magnitude compared to the serial version.

The content of this article is as follows. In Section 2, the basic characteristics of the N-MPCD algorithm will be described. Subsequently, in Section 3, the parallelization of the method on GPUs will be discussed. In Section 4, the main results of our research will be presented, and in Section 5, conclusions will be synthesized and possible future work will be proposed.

2 Simulation Method

The simulation system consists of point particles that move within a cubic box with side length L, which is considered an integer multiple of the unit length a. All particles have the same mass m. Their positions and velocities are represented by the vectors $\mathbf{r}_i$ and $\mathbf{v}_i$, with $i = 1, 2, \ldots, N$. Each particle has an associated unit orientation vector, $\hat{\mathbf{u}}_i$, which will serve to generate the characteristic orientation order of NLCs. The vectors $\mathbf{r}_i$, $\mathbf{v}_i$, and $\hat{\mathbf{u}}_i$ are considered continuous functions of time, t, and will be updated to generate system's dynamics. The algorithm responsible for this consists of two steps known as the propagation step and the collision step. Both will be described below.

2.1 Propagation Step

Particles move in uniform rectilinear motion for a fixed time interval Δt. This updates the position of each particle according to the equation

$$\mathbf{r}_i\left(t + \Delta t\right) = \mathbf{r}_i\left(t\right) + \mathbf{v}_i\left(t\right)\Delta t. \tag{1}$$

This displacement implies that some particles will leave the simulation box. To keep the number of particles constant and approximate macroscopic behavior, periodic boundary conditions are imposed. Thus, each particle that exits one side of the box is replaced by another that enters from the opposite side with the same velocity and orientation. This is achieved through the transformation

$$x_i \longrightarrow x_i - L\,\mathrm{floor}\left(\frac{x_i}{L}\right), \tag{2}$$

where the floor function, floor$\,(x)$, that returns the largest integer less than or equal to x, and x_i is the first Cartesian component of $\mathbf{r}_i$. A similar transformation applies to components y_i and z_i.

2.2 Collision Step

After propagation, particles are grouped into cubic cells of volume a^3, distributed in a cubic lattice which fills the entire simulation box. These cells are called *collision cells* because the particles that end up within the same cell participate in an exchange of velocities and orientations that is equivalent to a fictitious multiple collision among them. At every collision step the number of particles in each collision cell could be different since particles move from one collision cell to another during the propagation step. At any given instant, the physical fields of the system can be calculated using the particles located within each collision cell.

The new velocities are assigned using the Andersen thermostat rule [6],

$$\mathbf{v}_i\left(t+\Delta t\right)=\mathbf{v}^{\mathrm{c}}(t)+\boldsymbol{\xi}_i-\boldsymbol{\xi}^{\mathrm{c}}, \tag{3}$$

where

$$\mathbf{v}^{\mathrm{c}}(t)=\frac{1}{N^{\mathrm{c}}}\sum_{i=1}^{N^{\mathrm{c}}}\mathbf{v}_i, \tag{4}$$

is the center of mass velocity in the cell where the particle is located, with N^{c} being the number of particles in that cell.

In addition, in equation (3), $\boldsymbol{\xi}_i$ is a random contribution taken with probability

$$P\left(\boldsymbol{\xi}_i\right)=\left(\frac{m}{2\pi k_{\mathrm{B}}T}\right)^{\frac{3}{2}}\exp\left(-\frac{m}{2k_{\mathrm{B}}T}\boldsymbol{\xi}_i\cdot\boldsymbol{\xi}_i\right), \tag{5}$$

which corresponds to the velocities of molecules in a fluid at temperature T, with k_{B} being the Boltzmann constant.

In equation (3),

$$\boldsymbol{\xi}^{\mathrm{c}}=\frac{1}{N^{\mathrm{c}}}\sum_{i=1}^{N^{\mathrm{c}}}\boldsymbol{\xi}_i, \tag{6}$$

is a term that guarantees the local conservation of linear momentum after velocity update.

To update orientations, it is considered that the particles within the same collision cell interact with the director produced by themselves, $\hat{\mathbf{n}}^{\mathrm{c}}$. To select the new orientations of the particles in the collision cells, the probability distribution is considered [19]

$$P\left(\theta_i\right)\sin\theta_i\,d\theta_i=C_0\exp\left(\frac{3US^{\mathrm{c}}}{2k_{\mathrm{B}}T}\left(\cos^2\theta_i-\frac{1}{3}\right)\right)\sin\theta_i\,d\theta_i, \tag{7}$$

where C_0 is a normalization constant, θ_i is the angle between $\hat{\mathbf{u}}_i$ and $\hat{\mathbf{n}}^{\mathrm{c}}$, U is a scalar quantity referred to as the *nematicity*, which defines the order in the simulated phase, and S^{c} is the so-called order parameter in the cell, which is the largest eigenvalue of the tensor order parameter at the cell,

$$\mathbf{Q}^{\mathrm{c}}=\frac{1}{2N^{\mathrm{c}}}\sum_{i=1}^{N^{\mathrm{c}}}\left(3\hat{\mathbf{u}}_i\hat{\mathbf{u}}_i-\mathbf{I}\right). \tag{8}$$

S^c measures the amount of orientational order and takes two extreme values, $S^c = 0$, when the fluid is completely disordered, and $S^c = 1$, when the alignment of the particles is perfect. In addition, $\hat{\mathbf{n}}^c$ is the eigenvector of $\mathbf{Q}^c$ corresponding to S^c.

The probability density in equation (7) has the form of a canonical law based on the Maier-Saupe mean-field energy. It is known as the Dawson distribution and is illustrated in Figure 3 for different values of U. It can be seen that when U is small, the probability of the angles tends to be uniform, implying a disordered phase; while as U increases, the most probable angle is close to 0, indicating alignment of the particles around $\hat{\mathbf{n}}^c$.

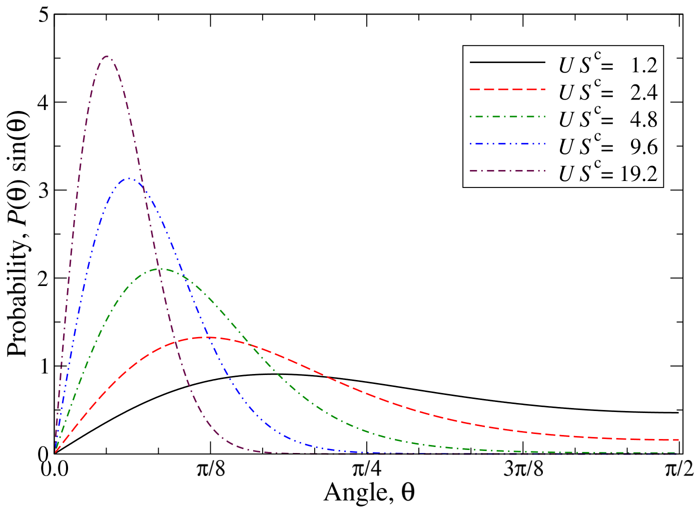


Fig. 3. Dawson distribution, equation (7), for diverse values of the quantity US^c normalized with respect to $k_B T$.

To validate the numerical implementation, the orientations of the resulting particles were taken in tests with different values of U. From these orientations, histograms were constructed that fit very well with the theoretical distribution given by equation (8), as shown in Figure 4.

3 Parallelization

3.1 General Considerations

The program was written in C and the CUDA API. Graphic cards were chosen instead of the machine's central processor because the former typically have more processing threads. The computing resources used for this research and the execution of numerical tests were: a computer with a Linux operating system,

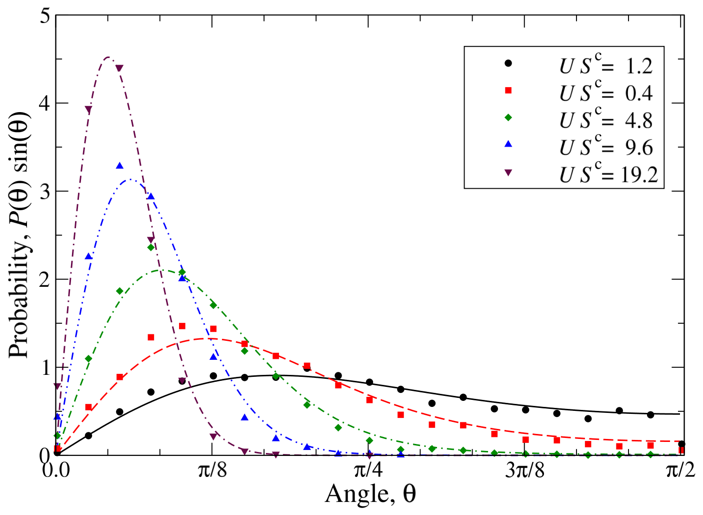


Fig. 4. Dawson distribution for diverse values of the quantity US^{c} normalized with respect to $k_{\mathrm{B}}T$. Curves are obtained from equation (7) whereas symbols correspond to normalized frequencies on samples of $81,920$ angles generated by the numerical implementation.

equipped with a Tesla T4 graphics card that has $2,560$ CUDA threads, and an Intel Xeon E5 2640 CPU with 16 processing threads and an x86_64 architecture.

When analyzing the various stages that comprise a simulation step in the N-MPCD method interval Δt, we can see that these fall into one of two categories:

1. exhaustive stages, where each particle is analyzed individually; and
2. summarized stages, where the system is analyzed at the collision cell level.

Armed with this information, we can see that we have two minimal computing units, the particle and the collision cell, depending on the simulation stage we are in. To maximize the amount of work carried out in parallel, the number of processing threads during the program execution is chosen to be equal to the minimal computing unit.

To store the processed information and avoid memory collisions, two large arrays of structures corresponding to the two minimal computing units were created. It is worth mentioning that older versions of CUDA do not support the use of double-precision floating-point variables. However, in our tests, we noticed that the truncation error produced by using float variables is too large to obtain reliable simulation results. Therefore, the produced code cannot be executed on older GPUs [20].

Another consideration to take into account is the limitations of the x86_64 computer architecture. In our case, we encountered two very important ones, one technical and one historical. The technical limitation is that in the x86_64 architecture, graphics memory is different from main memory, so all information to be computed on the GPUs must be transferred between them. The historical limitation has to do with the hardware restrictions that existed when the PC standard was proposed, as on certain equipment, the number of memory addresses available for the GPU is less than what would be necessary to index all the graphics memory to the computer's data bus.

To maintain compatibility with x86_64, NVIDIA GPUs do not expose all their memory to the data bus simultaneously. Instead, they expose only a small memory window, which can be shifted at the processor's request to allow reading and writing of the entire graphics memory. This procedure is schematically illustrated in Figure 5. While this solution allows graphics cards to have large amounts of memory, it makes the data transfer between RAM and graphics memory a slow process that consumes CPU time and, therefore, should be avoided as much as possible.

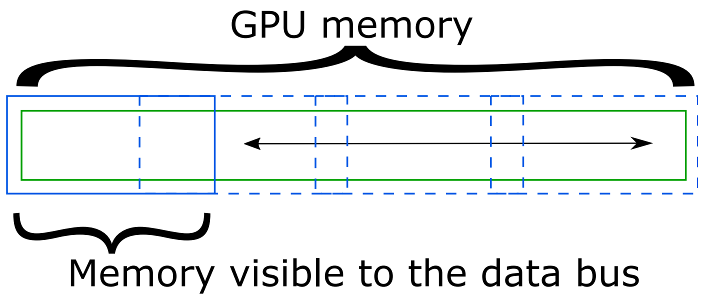


Fig. 5. Visual representation of the graphics memory in modern systems. The green rectangle represents all the graphics memory inside the GPU. The CPU can only read or write information inside the blue window. The CPU can move the window in order to write the whole graphics memory.

Lastly, it should be noted that all functions executed within the GPU must be of the void type, so data transfer and error conditions between functions must be handled via pointers [2]. Taking into account the general considerations above, it was decided to implement the parallelized N-MPCD algorithm according to the block diagram shown in Figure 5(a). For comparison purposes, the corresponding diagram for a serial version is also shown in Figure 5(b). The main steps of the method are detailed below.

3.2 Initialization

The initialization process is exhaustive, where each particle in the simulation system is randomly assigned initial values of position, orientation, and velocity. Fortunately, CUDA includes functions for generating random numbers using various distributions. For each numerical test, random number series were generated from seeds taken from the computer's clock.

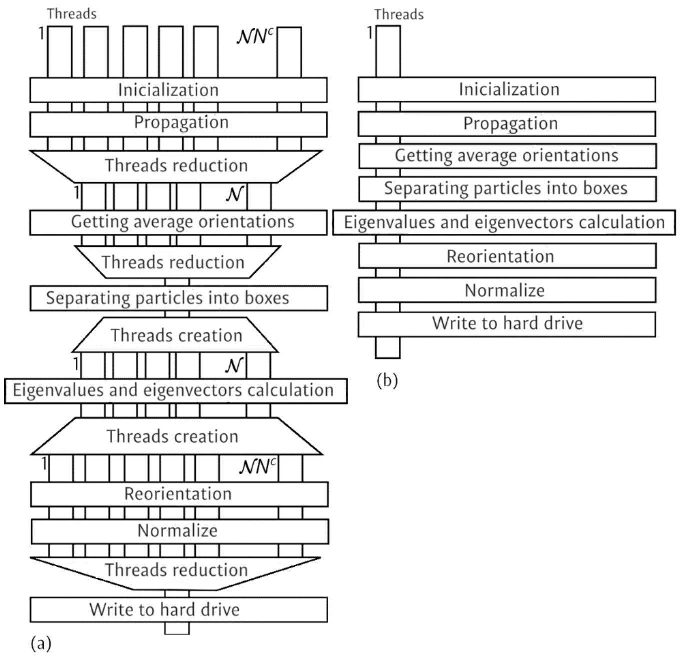


Fig. 6. Block diagram of the N-MPCD method, parallelized (a) and serial (b). The collision step in both cases goes from the calculation of the particles average orientation to the normalization. The serial algorithm runs this operations on a single processing thread.

3.3 Propagation

The movement stage remains an exhaustive process where the position of the particles within the simulation system is updated assuming uniform rectilinear motion. This is done by simply multiplying the velocity by Δt and adding the result to the current position of the particle for each of the Cartesian axes, as indicated by equation (1).

Boundary Conditions The N-MPCD method assumes that the simulation system is surrounded by identical copies of itself. Therefore, if a particle exits the system, it will be automatically replaced by an identical one from one of the adjacent systems. In practice, due to the finite nature of computing resources, the same particle is reintroduced into the system programmatically by implementing equation(2).

3.4 Collision

The collision stage, unlike the previous ones, is no longer exhaustive. To obtain the information that describes the current state of the collision cells, it is necessary to first average the orientation and velocity of all the particles within them. Ideally, while the number of processing threads remains equal to the number of simulated particles, each particle would add its own data to the average of its corresponding cell so that after thread reduction, each collision cell performs the final division. The problem with this implementation is that it inevitably leads to race conditions between the different processing threads writing data to the same variable. Traditionally, this would not be an issue as it is easily solved by implementing a semaphore [3]. Unfortunately, the current state of the CUDA API does not include semaphore functions.

To avoid race conditions, we chose to average the particle data after thread reduction, ensuring that each collision cell sums its own average. This way, we avoid having multiple threads attempting to write to the same variable. Although this implementation is not as efficient as the one previously described, it still performs parallel work, making it faster than a completely serial implementation.

Calculation of the Director Vector To reorient the particles within each collision cell, the N-MPCD method requires calculating the director vector around which the particles will rotate. This is clearly demonstrated by equation (7), which depends on the angle each particle forms with its local director.

As mentioned, the director is calculated by obtaining the eigenvalues and eigenvectors of the order parameter tensor, $\mathbf{Q}^{\mathrm{c}}$, which is defined by equation (8). The scalar order parameter in the cell, S^{c}, is the highest eigenvalue of $\mathbf{Q}^{\mathrm{c}}$, while the corresponding eigenvector is the local director $\hat{\mathbf{n}}^{\mathrm{c}}$.

As we can see, calculating $\mathbf{Q}^{\mathrm{c}}$ leads us once again to a race condition problem. In this specific case, CUDA provides a set of so-called atomic functions, which are designed to perform the four basic arithmetic operations in parallel for the different types of numerical variables that exist in the C language [2].

Although atomic functions do a good job of handling possible collisions between processing threads, in the numerical tests carried out in this project, it was noted that when the size of the simulated system exceeds 8^3 particles, the number of collisions becomes so large that the GPU has to dedicate a significant amount of time resolving them before it can perform the summation. In the long run, this causes the execution of the parallel program to be slower than the serial version. Therefore, we opted to perform this part of the program serially, once again choosing a suboptimal implementation but maintaining the integrity of the results.

Assignment of New Orientations Once the calculation of S^{c} and $\hat{\mathbf{n}}^{\mathrm{c}}$ is completed, we obtain new orientations for the particles from the Dawson distribution, equation (8). New velocities are also assigned to them from equations (4) to (6).

3.5 Writing to Disk

Finally, the obtained results are written to the hard disk. Due to physical constraints in the movement of read/write heads in most hard disks, this is not really a parallelizable process. However, it is possible to generate an additional processing thread that writes data to disk while the GPU continues executing the program.

4 Results

The numerical tests performed aim to establish the physical validity of the method and its performance compared to an existing serial version.

4.1 Nematic Behavior

The method allows observing an orientationally ordered phase for certain values of the nematicity, U. In a first set of simulations, tests were performed for a cubic system with side length $L = 32\,a$, where we maintained an average of 20 particles per collision cell. The average order parameter was calculated for different values of $U = 1, 2, 4, 8, 16, 20$, and 32 $k_{\mathrm{B}}T$.

The results obtained are illustrated in Figure 7, where a transition from disordered states, where $S^{\mathrm{c}} \sim 0$, to ordered states with $S \sim 0.8$ is observed. The former correspond to simple fluid phases and are obtained for $U \lesssim 5\,k_{\mathrm{B}}T$. The ordered states are observed when $U > 5\,k_{\mathrm{B}}T$ and correspond to nematic phases.

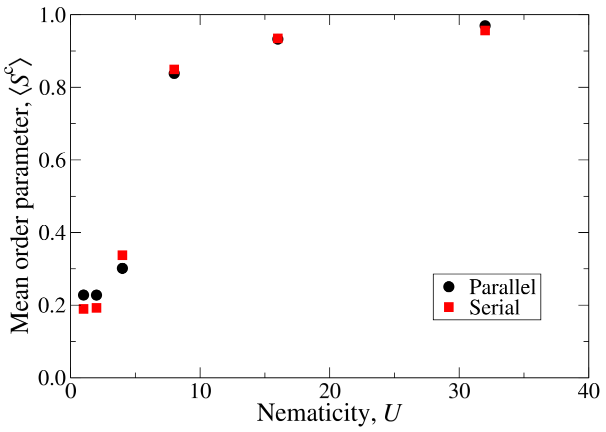


Fig. 7. Behavior of the orientational order of N-MPCD systemas as function of U. Results show that systems achieve order as U increases. A slight difference between results from the serial and parallel implementations can be observed, which is attributable to the numerical precision used in each case.

Another way to understand this behavior is through Figure 8, which illustrates the state of the simulated system using the GPU-parallelized code for different values of U. Two completely disordered states can be seen when $U = 1\,k_{\mathrm{B}}T$

and $U = 4\,k_{\mathrm{B}}T$, cases 7(a) and 7(b), respectively. On the other hand, the system acquires orientational order for values $U = 8\,k_{\mathrm{B}}T$ and $U = 16\,k_{\mathrm{B}}T$, cases 7(c) and 7(d), respectively.

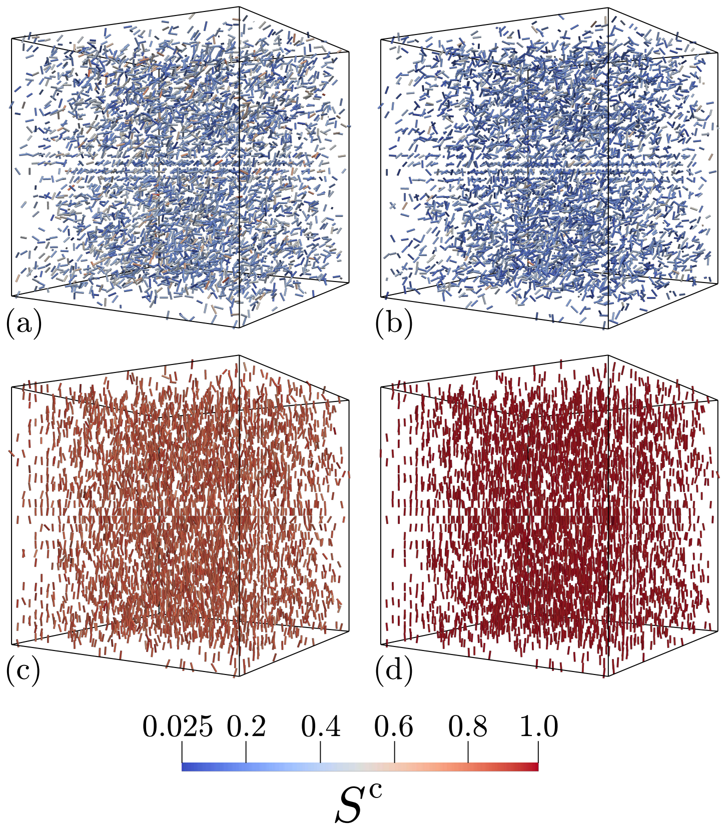


Fig. 8. Ordered and disordered phases simulated by N-MPCD parallelized in GPUs. (a) Disordered sate at $U = 1\,k_{\mathrm{B}}T$. (b) Disordered sate at $U = 4\,k_{\mathrm{B}}T$. (c) Nematic state at $U = 8\,k_{\mathrm{B}}T$. (d) Nematic state at $U = 16\,k_{\mathrm{B}}T$. Small bars indicate the local director field whereas the color scale at the bottom represents the local order parameter. Notice that orientation vectors correspond to local averages in the collision cells. They are not orientations of the individual particles of the method. This is why they are distributed over the same positions.

4.2 Performance

To estimate the performance of the developed implementation, we considered the computation time, as traditional performance comparison methods depend on the similarity in the architecture of the processors where the code is executed.

The first comparison was made between a series of simulations where the nematicity was modified, keeping the parameters fixed: $L = 32\,a$, 20 particles on average per collision cell, $\Delta t = 0.1$, and $k_{\rm B}T = 1.0$, and $m = 1$. The reported values are the result of a sample of 100 consecutive simulations for each nematicity value. The computation time shown in Table 1 is the average over these samples. It is worth mentioning that the computation times include contributions from writing to disk, which aims to save the information corresponding to the state of the system (value of the order parameter in each collision cell) after each step of the algorithm. This file writing can be considered optional as its purpose was to allow visualization of the system configuration.

Table 1. Average computing times of the parallel N-MPCD method, using different values of U whit parameters $N^c = 20$, $\Delta t = 0.1$, $k_{\rm B}T = 1$ y $m = 1$.

Nematicity (U)	**Computing time (h:m:s)**
1	00:02:15
2	00:02:15
8	00:02:20
16	00:02:17
32	00:02:17

Execution times were also calculated for systems of different sizes with 20 particles on average per collision cell and fixed $U = 1\,k_{\rm B}T$. The results obtained in this case are shown in Table 2 for the serial implementation of the N-MPCD method, whereas Table 3 reports on simulation times obtained with the parallel implementation. Notice that the number of collision cells used to assess the performance of the method varied from $8^3 = 512$ to $64^3 = 262\,144$, whereas the total number of simulated particles varied from 10 240 to 5 242 880, respectively.

Table 2. Average computing times of a serial implementation of the N-MPCD method for different values of N, with parameters $U = 1$, $N^c = 20$,$\Delta t = 0.1$, $k_{\rm B}T = 1$ and $m = 1$.

System size (N)	**Computing time (h:m:s)**
8^3	00:00:15
16^3	00:01:32
24^3	00:05:39
32^3	00:07:49
48^3	00:27:05
64^3	01:25:01

It was observed that the performance of the parallel algorithm is much superior to its serial equivalent. This behavior is emphasized when simulation times

Table 3. Average computing times of a parallel implementation of the N-MPCD method for different values of N, with parameters $U = 1$, $N^c = 20$,$\Delta t = 0.1$, $k_BT = 1$ and $m = 1$.

System size (N)	Computing time (h:m:s)
8^3	00:00:02
16^3	00:00:16
24^3	00:00:46
32^3	00:02:15
48^3	00:05:49
64^3	00:12:53

are compared graphically as in Figure 4.2. In the most significant cases, the system sizes were $L = 48\,a$ and $L = 64\,a$, with the reduction in computation time achieved by the parallelized method being approximately 80% and 85%, respectively. In this regard, it can be concluded that our implementation constitutes a solid first step in the development of a liquid crystal simulation method that, in the near future, could serve as a robust tool for the analysis of such systems.

Since the method developed in this work deals with non-sequential programming, it is convenient to show the algorithm's scaling, also known as speedup, as well as its efficiency. On the one hand, speedup is defined as the ratio of serial to parallel computation times. On the other hand, efficiency is defined as the ratio of speedup to the number of processing threads. The precise values of speedup and efficiency of the method developed in this research are shown in Table 4, which confirms the high performance achieved by the GPU-parallelized implementation.

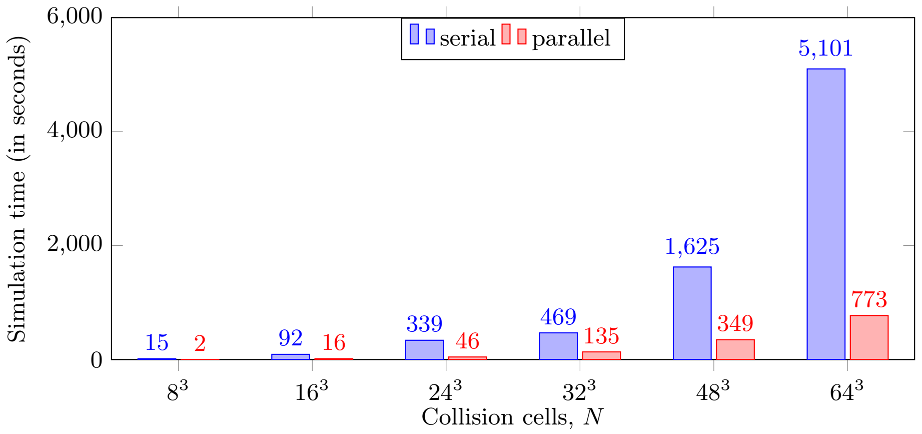


Fig. 9. Comparison between simulation times of the parallel and serial programs (blue and red bars, respectively) for different system sizes quantified by the total number of collision cells.

Table 4. Speedup and efficiency of the parallel implementation of the N-MPCD method

System size (N)	Speedup	Efficiency
8^3	2.17	4.0×10^{-3}
16^3	1.49	3.6×10^{-4}
24^3	2.30	1.6×10^{-4}
32^3	3.12	9.5×10^{-5}
48^3	4.76	4.3×10^{-5}
64^3	41.67	1.5×10^{-4}

5 Conclusions

In this research work, a GPU-parallelized algorithm was implemented to simulate nematic liquid crystals (NLC). The approach used to reproduce the physical characteristics of the nematic phase was inspired by the Nematic Multiparticle Collision Dynamics (N-MPCD) method, which combines particle movements with collision rules to satisfy equilibrium conditions and control the orientational order characteristic of NLCs. Collisions between particles occur in limited and independent spatial regions, allowing operations to be distributed in parallel. The parallelized development was based on NVIDIA technology. The simulations were executed on Tesla T4 cards with 2,560 processing threads, on a computer with a Xeon E5 CPU and x86_64 architecture.

The advantages of the developed code include a significant reduction in computation time and the ability to simulate systems with more particles compared to a serial version of the algorithm. Specifically, the numerical tests performed resulted in a reduction of computation time by an order of magnitude when compared to tests of a serial implementation. Though, in principles, this reduction has to be compared with the two orders of magnitude gain reported for other MPCD implementations based on GPUs [23], it has to be stressed that such implementations do not simulate fluids with nematic features. In addition, it is worth emphasizing that using a GPU with more processing threads could be expected to result in even greater time reductions. Notably, even in the largest systems, GPU memory was not an issue during the method's execution.

Due to the complexity of the dynamics of liquid crystals, the code does not incorporate some of the steps proposed in the original N-MPCD algorithm referenced in [9]. Our implementation does not include the coupling steps between flow and orientation variables. This coupling refers to the fact that in a real liquid crystal, the flow can induce director reorientation and a change in molecular orientation can induce flow.

Flow-induced reorientation can be incorporated in terms of the spatial derivatives of fluid velocity, which are estimated using finite differences between the velocities of different collision cells. These spatial changes in velocity impose torques on the director in each cell, thus causing reorientation. On the other hand, flow induced by reorientation is incorporated by transforming the angular momentum gain generated by reorientation into orbital angular momentum. Both mechanisms require summarized and extended operations whose parallel implementation is under development.

For future work, it is proposed to complement the method with external forces, such as electric fields or flows, to explore the algorithm's capability to reproduce more complex situations, making it a reliable predictive tool for the behavior of liquid crystals. Additionally, it is recommended to develop a graphical interface that allows for user-friendly manipulation of simulation parameters and to create versions of the algorithm that can run on other hardware platforms not limited to NVIDIA technology.

References

1. Care, C., Cleaver, D.: Computer simulation of liquid crystals. Reports on progress in physics 68(11), 2665 (2005)
2. Cook, S.: CUDA programming: a developer's guide to parallel computing with GPUs. Morgan Kaufmann, Waltham (2012)
3. Downey, A.: The Little Book of Semaphores. Green Tea Press, Massachusetts (2016)
4. Dunmur, D., Sluckin, T.: Soap, science, and flat-screen TVs: a history of liquid crystals. Oxford University Press, Oxford (2014)
5. Feynman, R., Sands, M., Leighton, R.: The Feynman Lectures on Physics, vol. II. Basic Books, New York (2011)
6. Gompper, G., Ihle, T., Kroll, D.M., Winkler, R.G.: Multi-particle collision dynamics – a particle-based mesoscale simulation approach to the hydrodynamics of complex fluids. In: Holm, C., Kremer, K. (eds.) Advanced Computer Simulation Approaches for Soft Matter Sciences III, vol. 221, p. 1–87. Springer, Berlin, Heidelberg (2009)
7. Halver, R., Junghans, C., Sutmann, G.: Using heterogeneous gpu nodes with a cabana-based implementation of mpcd. Parallel Computing 117, 103033 (2023), https://www.sciencedirect.com/science/article/pii/S016781912300039X
8. Híjar, H.: Hydrodynamic correlations in isotropic fluids and liquid crystals simulated by multi-particle collision dynamics. Condens. Matter Phys. 22(1), 13601 (2019)
9. Híjar, H.: Curso introductorio de cristales líquidos i: fases y propiedades estructurales. Revista Mexicana de Física E (2024), in press, also at https://doi.org/10.48550/arXiv.2403.03366
10. Híjar, H., Halver, R., Sutmann, G.: Spontaneous fluctuations in mesoscopic simulations of nematic liquid crystals. Fluct. Noise Lett. 18(3), 1950011 (2019)
11. Hirst, L.: Fundamentals of Soft Matter Science. CRC Press, Boca Raton (2012)
12. Howard, M.P., Nikoubashman, A., Palmer, J.C.: Modeling hydrodynamic interactions in soft materials with multiparticle collision dynamics. Current Opinion in Chemical Engineering 23, 34–43 (2019), https://www.sciencedirect.com/science/article/pii/S2211339819300024, frontiers of Chemical Engineering: Molecular Modeling
13. Howard, M.P., Panagiotopoulos, A.Z., Nikoubashman, A.: Efficient mesoscale hydrodynamics: Multiparticle collision dynamics with massively parallel

gpu acceleration. Computer Physics Communications 230, 10–20 (2018), https://www.sciencedirect.com/science/article/pii/S0010465518301218

14. Lee, K.W., Mazza, M.G.: Stochastic rotation dynamics for nematic liquid crystals. J. Chem. Phys. 142(16), 164110 (2015)
15. Palffy-Muhoray, P.: The diverse world of liquid crystals. Phys. Today 60, 54–60 (2007)
16. Peng, C., Turiv, T., Guo, Y., Wei, Q.H., Lavrentovich, O.D.: Command of active matter by topological defects and patterns. Science 354, 882–885 (2016)
17. Petersen, M., Lechman, J., Plimpton, S., Grest, G., Veld, P., Schunk, P.: Mesoscale hydrodynamics via stochastic rotation dynamics: Comparison with lennard-jones fluid. J.Chem.Phys 132, 174106 (2010)
18. Ratan, S.S.: Gpu-based multiscale simulation to model active matter hydrodynamics in fluid medium (2023), bS-MS Thesis. Indian Institute of Science Education and Research Pune
19. Shendruk, T.N., Yeomans, J.M.: Multi-particle collision dynamics algorithm for nematic fluids. Soft Matter 11, 5101–5110 (2015)
20. Soyata, T.: GPU Parallel Program Development Using CUDA. CRC Press, Boca raton (2018)
21. Tomilin, M.G., Povzun, S.A., Kurmashev, A.F., Gribanova, E.V., Efimova, T.A.: The application of nematic liquid crystals for objective microscopic diagnosis of cancer. Liq. Cryst. Today 10, 3–5 (2001)
22. Wang, H., Xu, T., Fu, Y., Wang, Z., Leeson, M., Jiang, J., Liu, T.: Liquid crystal biosensors: Principles, structure and applications. Biosensors 12, 639–666 (2022)
23. Westphal, E., Singh, S., Huang, C., Gompper, G., Winkler, R.: Multiparticle collision dynamics: Gpu accelerated particle-based mesoscale hydrodynamic simulations. Comput.Phys.Commun 185, 495–503 (2014)

ISSN 1870-4069

Categorización de tipos de jugador para una estrategia didáctica gamificada entre estudiantes de ingeniería

Miguel Ángel Guzmán Rivera[1], Sandra Luz Canchola-Magdaleno[2], Ma. Elena Montes Almanza[1], María Luisa Montes Almanza[1], Víctor Alejandro González-Huitrón[1], Ma. Del Consuelo Frías Maldonado[1]

[1] Tecnológico Nacional de México / Instituto Tecnológico de Querétaro, Departamento de Sistemas y Computación, México

[2] Universidad Autónoma de Querétaro, Facultad de Informática, Departamento de Posgrado, México

miguel.gr@queretaro.tecnm.mx

Resumen. La familiaridad que tienen los estudiantes en la actualidad con las tecnologías digitales impone cambios en las estrategias didácticas para la enseñanza de las ciencias, tecnología, ingeniería y matemáticas en la educación superior. Una de las estrategias más prometedoras es la gamificación, pues puede mejorar la motivación para lograr un aprendizaje significativo, y puede incorporar con facilidad el uso de herramientas digitales. Para determinar el tipo de jugador prevalente entre estudiantes de ingeniería se aplicó un test de Bartle entre una muestra representativa de 71 participantes, obteniendo una distribución uniforme entre cuatro categorías de tipo de jugador, con tres características destacadas en común. Esta información servirá para seleccionar mecánicas de juego apropiadas para implementar estrategias gamificadas para el aprendizaje de competencias de programación.

Palabras clave: Gamificación, categorización de jugadores, STEM, educación superior.

Categorization of Player Types for a Gamified Teaching Strategy among Engineering Students

Abstract: The familiarity that students currently have with digital technologies imposes changes in teaching strategies for teaching science, technology, engineering and mathematics in higher education. One of the most promising strategies is gamification, as it can improve motivation to achieve meaningful learning, and can easily incorporate the use of digital tools. To determine the type of player prevalent among engineering students, a Bartle test was applied among a representative sample of 71 participants, obtaining a uniform distribution between four categories of player type, with three outstanding characteristics in common. This information will be used to select appropriate game mechanics to implement gamified strategies for learning programming skills.

Keywords: Gamification, player categorization, STEM, higher education.

1. Introducción

La educación en ciencias, tecnología, ingeniería y matemáticas (STEM, por sus siglas en inglés) se ha convertido en un pilar fundamental para el desarrollo económico en la era contemporánea. En primer lugar, la formación en áreas STEM proporciona a los individuos habilidades críticas y analíticas esenciales para enfrentar los desafíos tecnológicos y científicos del siglo XXI. La capacidad de resolver problemas complejos, el pensamiento crítico y la innovación son competencias que se cultivan a través de la educación STEM y son indispensables para el avance en una economía global basada en el conocimiento [1].

Es por este motivo que la inmersión en la educación STEM, particularmente a nivel universitario, es un motor clave para la competitividad económica de los países, al contribuir a la creación de una fuerza laboral altamente calificada que puede adaptarse rápidamente a las demandas cambiantes del mercado laboral y las innovaciones tecnológicas. Las sociedades que fomentan y fortalecen la educación en estos campos son más capaces de desarrollar industrias tecnológicamente avanzadas, impulsar la investigación y el desarrollo, así como atraer mayores inversiones. Esto, a su vez, fomenta la generación de empleos y promueve un crecimiento económico sostenible [2].

La educación superior se enfrenta, sin embargo, a desafíos que enfatizan la necesidad de reconsiderar los métodos tradicionales de instrucción debido a la creciente alfabetización digital de los estudiantes. Tradicionalmente, la educación ha sido conceptualizada como un proceso de instrucción presencial con el objetivo de apoyar y desarrollar al máximo la personalidad y el potencial natural de los estudiantes. Este ideal educativo no ha cambiado en la actualidad, aunque los medios y canales de diseño del proceso educativo han tenido que adaptarse acorde a los cambios sociales y tecnológicos, para satisfacer las expectativas de los nativos digitales que ocupan las aulas hoy en día [3].

Debe tomarse en consideración que, si bien los alumnos en la actualidad pueden ser considerados como nativos digitales en referencia al manejo de dispositivos electrónicos tales como smartphones, tablets, laptops, etc., esto no significa necesariamente que se adaptarán naturalmente a cambios en el modelo educativo que involucren dichas tecnologías. Sin embargo, con un planteamiento didáctico adecuado, es perfectamente viable instrumentar actividades de aprendizaje significativo en las aulas de la educación superior para la adquisición de competencias relacionadas con las STEM [4], mediante el uso de las herramientas digitales a las que tienen acceso comúnmente los estudiantes.

2. Antecedentes

En la actualidad los estudiantes manifiestan diferentes patrones de pensamiento y procesan la información de manera diferente a generaciones anteriores en virtud de su familiaridad con el entorno digital de las computadoras, Internet y los videojuegos. Estos nativos digitales buscan, interactúan, crean, aprenden y socializan de formas diferentes [5], debido a toda una vida de exposición a las tecnologías de la información y comunicaciones (TIC).

Es por esta razón que, en la enseñanza de la ingeniería, es necesario el uso de estrategias didácticas en las instituciones de educación superior, que tomen ventaja de las herramientas digitales que ofrecen las TIC para mejorar la experiencia educativa y los resultados de aprendizaje. Estas herramientas digitales permiten la creación de entornos de aprendizaje interactivos y dinámicos que pueden simular situaciones reales de ingeniería. Por ejemplo, software de simulación y modelado 3D permiten a los estudiantes experimentar y visualizar conceptos complejos, como la dinámica de fluidos, el comportamiento estructural o los circuitos eléctricos, en un entorno controlado y seguro. Estas simulaciones facilitan la comprensión y aplicación práctica de teorías abstractas, mejorando la retención y la capacidad de resolución de problemas.

También debe considerarse el uso de plataformas digitales de aprendizaje, tales como los sistemas de gestión del aprendizaje (LMS), que ofrecen recursos y materiales didácticos accesibles en cualquier momento y lugar, lo que fomenta el aprendizaje autónomo y flexible. Los estudiantes pueden acceder a conferencias grabadas, tutoriales en video, artículos académicos y ejercicios interactivos, lo que les permite aprender a su propio ritmo y profundizar en los temas según sus necesidades e intereses. Además, estas plataformas suelen incluir herramientas de evaluación automatizadas, que proporcionan retroalimentación inmediata, ayudando a los estudiantes a identificar sus fortalezas y áreas de mejora de manera oportuna.

Una estrategia didáctica que resulta atractiva a los nativos digitales y que toma provecho de las plataformas digitales es la gamificación, entendida esta como el uso de elementos y principios de diseño de juegos en contextos educativos con el propósito de mejorar la motivación, el compromiso y el aprendizaje de los estudiantes. Esta metodología se basa en la incorporación de dinámicas de juego, tales como la obtención de puntos, niveles, recompensas, desafíos y competencias, en el entorno educativo. Al hacer esto, se busca crear experiencias de aprendizaje más atractivas y dinámicas, que fomenten la participación activa y la persistencia de los alumnos en la realización de tareas académicas. El presente estudio tiene como objetivo investigar el tipo de jugador prevalente entre los estudiantes de ingeniería para mejorar las estrategias gamificadas en la educación STEM.

La relación de la gamificación con las herramientas digitales es intrínseca y fundamental para su implementación efectiva, y ha sido implementada con éxito en la enseñanza de la ingeniería [6]. Las plataformas digitales permiten la creación de entornos interactivos y personalizados donde los elementos de gamificación pueden ser fácilmente integrados y gestionados. Por ejemplo, aplicaciones educativas y sistemas de gestión del aprendizaje pueden incluir características como insignias, tablas de clasificación y misiones, que incentivan a los estudiantes a participar y alcanzar metas de manera lúdica. Además, las herramientas de las TIC facilitan el seguimiento y análisis del progreso de los estudiantes, proporcionando datos valiosos que pueden ser

Tabla 1. Mecánicas de la gamificación.

Mecánica de juego	Descripción
Puntos	Recompensas virtuales por el esfuerzo del jugador. Son la unidad granular de medida en la gamificación
Logros	Completar metas específicas planteadas por el juego
Tableros de liderazgo	Despliegue visual de comparación social, basado en puntos y logros
Insignias	Visualización de los logros del jugador
Grafo social	Representación de la red social del jugador. Las relaciones entre participantes son un importante factor motivacional
Enfrentamientos con jefes	Retos especiales al final de cada nivel
Colecciones	Conjunto de objetos virtuales acumulados
Retos	Objetivos planteados para lograr la motivación del jugador
Desbloqueo de contenidos	Privilegio para los jugadores al conseguir logros
Restricciones	Limitantes al uso de tiempo y de recursos que promueven la automotivación del jugador
Niveles	Progreso del jugador, presentado como una jornada personalizada
Avatares	Visualización del personaje del jugador
Misiones	Retos predefinidos con un objetivo específico
Narrativa	Planteamiento de retos y objetivos en forma de una historia dentro de un contexto que involucra emocionalmente al jugador
Equipo	Grupo de jugadores con una meta común para promover el aprendizaje colaborativo
Bienes virtuales	Recursos utilizables en el juego, resultado de conseguir puntos y logros

utilizados para ajustar y mejorar continuamente las estrategias de enseñanza, y les resultan familiares a los nativos digitales.

La gamificación se caracteriza por ser una técnica que emplea mecánicas de juego en entornos no lúdicos con el propósito de mejorar el compromiso de los usuarios con un servicio. Dichas mecánicas de juego son constructos formados por reglas y lazos de retroalimentación, y su aplicación exitosa dependerá de una estrategia didáctica de gamificación bien diseñada, construida con base en un adecuado entendimiento del participante, su misión y la motivación que lo impulsa. La intención de los autores del presente estudio es instrumentar actividades de aprendizajes gamificadas para la enseñanza de competencias de programación en carreras de ingeniería, tales como exploración de código en mazmorras, juegos de rol en un entorno de desarrollo de software, o sistemas de insignias y logros. Dichas estrategias han demostrado su viabilidad en la enseñanza de la ingeniería de software [7].

La Tabla 1 recopila las mecánicas de juego más comunes a la gamificación [8], las cuales pueden ser usadas de forma individual o combinada.

3. Método

Dada la necesidad de entender las motivaciones de los estudiantes que participarán en el entorno gamificado de aprendizaje, el primer paso es determinar la categoría de jugadores a la que estos pertenecen, estableciendo las siguientes preguntas de investigación:

PI1: *¿Cuál es el tipo de jugador más prevalente entre los estudiantes de Ingeniería en Sistemas Computacionales?*

PI2: *¿Cuál es el perfil motivacional del tipo de jugador más prevalente encontrado?*

Hasta ahora se han propuesto varias teorías que intentan describir diversos perfiles motivacionales de los jugadores. Una teoría prevalente propuesta por Bartle [9] la cual identifica "Tipos" de jugadores en función de su actividad preferida mientras juegan. Para identificar estos tipos, dicho autor desarrolló una prueba pionera en los estudios de juegos y mundos virtuales.

El test de Bartle es un instrumento de evaluación diseñado para clasificar a los jugadores de videojuegos en cuatro tipos principales, basándose en sus preferencias y comportamientos dentro del juego. Los cuatro tipos de jugadores identificados por Bartle son:

- Asesinos, quienes disfrutan de la competencia y el conflicto directo con otros jugadores.
- Exploradores, cuyo propósito es descubrir y conocer más sobre el entorno del juego.
- Socializadores, los cuales buscan la interacción y comunicación con otros jugadores.
- Ambiciosos, centrados en acumular puntos, subir de nivel y lograr objetivos específicos dentro del juego.

La relación del test de Bartle con la gamificación radica en su capacidad para proporcionar información valiosa sobre las motivaciones y preferencias de los usuarios, información que puede ser utilizada para diseñar sistemas de gamificación más efectivos [10]. Conociendo a qué tipo de jugador pertenece un usuario, los diseñadores pueden adaptar las mecánicas y dinámicas del sistema gamificado para satisfacer mejor sus motivaciones, ya sea a través de desafíos competitivos, oportunidades de exploración, interacciones sociales o recompensas y logros.

El método aplicado en esta investigación consistió en aplicar el test de Bartle, que consiste en 35 reactivos binarios, a un conjunto de 71 estudiantes que conforman una muestra representativa de alumnos de la carrera de Ingeniería en Sistemas Computacionales, con el fin de conocer el tipo de jugador más prevalente entre ellos, y en base a este conocimiento seleccionar las mecánicas de juego más apropiadas para implementar una estrategia de gamificación que mejore la motivación para lograr un aprendizaje significativo entre los estudiantes, de acuerdo con la Tabla 1.

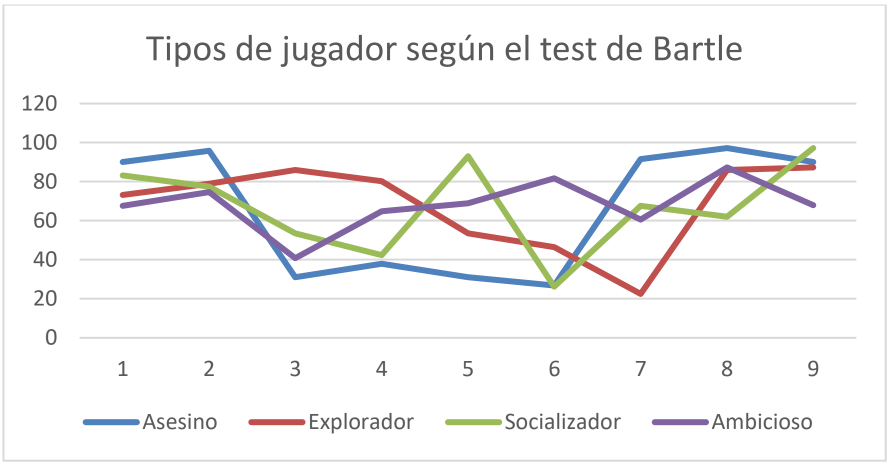


Fig. 1. Porcentaje de respuestas afirmativas a cada uno de los nueve reactivos planteados para cada tipo de jugador.

4. Resultados

Los resultados obtenidos tras la aplicación de la prueba para determinar el tipo de jugador manifiestan que el promedio de las cuatro categorías presenta una distribución uniforme, arrojando los siguientes datos:

– Tipo asesino: 66%,

– Tipo explorador: 68%,

– Tipo socializador: 67%,

– Tipo ambicioso: 68%.

Estos resultados, sin embargo, son engañosos. Aunque la media es bastante similar entre las categorías, al visualizar los datos de los reactivos individuales (ver Fig.1), queda de manifiesto que existen picos claramente distinguibles en varias de las preguntas planteadas.

Al seleccionar los reactivos que obtuvieron un porcentaje de respuesta afirmativa superior al 90%, surge una tendencia marcada hacia tres características comunes entre ellos, en donde los estudiantes manifiestan que:

– Se sienten atraídos por actividades que representen un reto a sus habilidades.

– Buscan la obtención de recompensas que representen un reconocimiento a sus logros.

– Son motivados por la competencia que representan los demás jugadores.

5. Conclusiones

Los resultados obtenidos pueden servir como guía para determinar el perfil motivacional de los estudiantes, lo que determinará las mecánicas de juego que llevarán a la mayor motivación para lograr los objetivos de aprendizaje establecidos en las actividades gamificadas propuestas. Seleccionar las mecánicas específicas dependerá, entre otros factores, de la plataforma de gestión del aprendizaje utilizada, de los criterios de evaluación para las asignaturas, así como del tema en particular que se pretende aprender.

Para ejemplificar, en el caso de un sistema gamificado de aprendizaje de competencias de programación para ingeniería, sabiendo que los estudiantes buscan retos a sus habilidades, disfrutan la competencia con sus compañeros y gustan de acumular recompensas, y asumiendo el uso de una plataforma de gestión de aprendizaje popular como Moodle, una buena selección de mecánicas aplicables podría ser la asignación de insignias por logros, desbloqueo de contenido por niveles y la publicación de un tablero de liderazgo grupal, todo lo cual puede instrumentarse con facilidad y puede, potencialmente, promover un aprendizaje significativo de las STEM al mejorar la motivación de los participantes [11].

Referencias

1. Agasisti, T., Bertoletti, A.: Higher Education and Economic Growth: A Longitudinal Study of European Regions 2000–2017. Socio-Economic Planning Sciences, 81 (2022) doi: 10.1016/j.seps.2020.100940.
2. Valero, A., Van Reenen, J.: The Economic Impact of Universities: Evidence from Across the Globe, Economics of Education Review. Elsevier, 68(C), pp. 53‒67 (2019) doi: 10. 1016/j.econedurev.2018.09.001.
3. Copaci, I., Rusu, A.: A Profile Outline of Higher Education E-Tutoring Programs for the Digital-Native Student – Literature Review. Procedia - Social and Behavioral Sciences, 209, pp. 145‒153 (2015) doi: 10.1016/j.sbspro.2015.11.270.
4. Córdoba-Valdés, F., Ramírez-Hernández, C., Sánchez-Guzmán, D.: Kinematics Learning in Engineering Students through Low-Cost Prototypes and 3d Printing. Research in Computing Science, 153(2) (2024)
5. Prensky, M.: The Emerging Online Life of the Digital Native: What they do Differently Because of Technology and How They do it (2004) http://www.bu.edu/ssw/files/pdf/ Prensky-The_Emerging_Online_Life_of_the_Digital_Native-033.pdf.
6. Díaz-Ramírez, J.: Gamification in Engineering Education – An empirical Assessment on Learning and Game Performance. Heliyon, 6(9) (2020) doi: 10.1016/j.heliyon.2020. e04972.
7. Porto, D., Jesus, G., Ferrari, F.: Initiatives and Challenges of Using Gamification in Software Engineering: A Systematic Mapping. Journal of Systems and Software, 173 (2021) doi: 10.1016/j.jss.2020.110870.
8. Kumar, J.: Gamification at Work: Designing Engaging Business Software. In: Proceedings of the Second International Conference on Design, User Experience, and Usability: Health, Learning, Playing, Cultural, and Cross-Cultural User Experience, pp. 528‒537 (2013) doi: 10.1007/978-3-642-39241-2_58.
9. Bartle, R.: Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs. Journal of MUD Research, 1(1), pp. 19 (1996)

10. Rowicka, M., Postek, S.: Who Likes to Learn New Things? How Gamification User Types and Satisfaction But Not the Frustration of Basic Psychological Needs Explain the Preference for Learning New Things. Acta Psychologica, 236 (2023) doi: 10.1016/j.actpsy.2023.103925.
11. Guzmán_Rivera, M., Escudero-Nahón, A., Canchola-Magdaleno, S.: "Gamificación" de la enseñanza para ciencia, tecnología, ingeniería y matemáticas: Cartografía conceptual. Sinéctica, 54 (2020) doi: 10.31391/s2007-7033(2020)0054-002.

A k-center-based Two-Phase Constructive Heuristic for the Depot-Free Multiple Traveling Salesperson Problem

ISSN 1870-4069

José Alejandro Cornejo-Acosta[1,2], Jesús García-Díaz[2,3], Blanca Verónica Zuñiga-Núñez[1]

[1] Tecnológico Nacional de México/ITS de Purísima del Rincón, Mexico

[2] Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Puebla, Mexico

[3] Consejo Nacional de Humanidades, Ciencias y Tecnologías, Mexico

{alejandro.ca, blanca.zn}@purisima.tecnm.mx, jesus.garcia@conahcyt.mx

Abstract. The Depot-Free Multiple Traveling Salesperson Problem (DFmTSP) is a variant of the classic Multiple Traveling Salesperson Problem (mTSP). In general, the purpose of the DFmTSP is that m salespersons must visit all the vertices of a given input complete weighted graph $G = (V, E, w)$ by minimizing an objective function. It has many applications in network optimization, routing, and logistics. Its main difference from other similar routing problems is that depots are not considered. In this paper, we introduce a Two-Phase constructive heuristic that uses an algorithm for the capacitated vertex k-center problem (CVKCP) in the first phase. For the second phase, a state-of-the-art heuristic for the classic TSP is used. The performed empirical evaluation shows that the proposal is capable of finding feasible and good-quality solutions in comparison to elaborated metaheuristics of the literature. Even more important, the proposal outperforms a novel metaheuristic when a percentage of imbalance between the number of vertices in the salespersons' paths is considered. Besides, one of the main advantages of the proposal is that it can find solutions in practical running times, which may be an important feature in certain situations.

Keywords: Network optimization, routing, heuristics, k-center, mTSP.

1 Introduction

Multiple Traveling Salesperson Problems (mTSPs) are a family of $\mathcal{NP}$-hard combinatorial optimization routing problems that generalize the classical Traveling Salesperson Problem (TSP). The mTSPs receive a complete weighted graph $G = (V, E, w)$ and a positive integer m as input. The purpose is to look for m paths for the salespersons that visit all the vertices in $V(G)$ by minimizing an objective function associated with the costs of the edges $E(G)$ [8]. In general, mTSPs can be applied in many routing and scheduling contexts, such as submarine patrol routing, bus routing, supervisor allocation, and some variants of job scheduling problems. Among the most popular studied variants of mTSP, two categories stand out:

- mTSPs that consider depots as part of the problem [12]:
  - SmTSP: All the salespersons must start their paths from a single defined vertex known as "depot".
  - MmTSP: There are multiple defined depots.
- mTSPs that do not consider depots at all [8]. These are known in the literature as Depot-Free mTSPs (DFmTSP).

Besides, mTSPs can also be classified according to the nature of the paths of the salespersons:

- Closed paths mTSPs (CP-mTSPs).
- Open paths mTSPs (OP-mTSPs).

In this context, a path is said to be closed if each salesperson starts and finishes its path at the same vertex. Otherwise, the path is said to be open. Among these mTSPs variants, DFmTSPs have received less attention in the literature in comparison with others, although they are considered one of the most fundamental variants [8]. This work focuses on the CP-DFmTSP. Besides, additional constraints to the maximum number of vertices each salesperson can visit are also considered. These are known in the literature as bounding constraints [12].

The rest of the paper is organized as follows: Section 2 reviews the related work of mTSPs and emphasizes the work focused on the DFmTSPs. Section 3 describes the proposal, a Two-Phase heuristic that uses the Cluster-First Route-Second strategy. Section 4 presents the carried out computational experimentation and performs an analysis of the results. Also, it discusses the advantages and disadvantages of the proposal. Finally, Section 5 states the conclusions and future work.

2 Literature Review

In general, mTSPs have been tackled through various optimization techniques, such as integer programming, approximation algorithms, exact algorithms, and heuristics and metaheuristics proposals. The initial integer programs (IPs) were proposed between 1960 and 1976 [16, 22, 9]. Over time, these IPs have been extended and improved to include considerations to address more realistic scenarios. [12] presented bounding constraints for these IPs for the SmTSP and MmTSP cases. In the context of the mTSP, bounding constraints specify that each salesperson must visit a minimum and/or maximum number of vertices. In the literature, most of the IPs have been proposed for mTSPs that consider depots as part of the problem. Nevertheless, recent advances in the literature have presented integer programs that are capable of dealing with both: mTSPs that consider depots and mTSPs that do not, and a combination between them [8]. Other recent proposals include the study of polyhedral approaches and branch-and-cut algorithms [2].

Besides IPs, other alternatives have been explored to approach mTSPs. As previously mentioned, mTSPs are $\mathcal{NP}$-hard. For this reason, in the last years, many researchers have proposed heuristic and metaheuristic algorithms in order to try to solve relatively big instances in practical running times. Among all the heuristics and metaheuristics proposed for mTSPs, evolutionary computing and some of their variants stand out [3, 26, 13].

Some recent proposals for the mTSP consist of hybridizing genetic algorithms with other metaheuristic algorithms. Such is the case of [11], where a genetic algorithm and an ant algorithm are combined, and [24], where a genetic algorithm is integrated with an invasive weed algorithm (IWO). Further, variants of ant algorithms have been used for similar routing problems like the mTSP with capacity and time windows [23] and the multi-objective Green Vehicle Routing Problem [15]. Other approaches that have been explored for mTSPs, are Two-Phase heuristics, which, as the name suggests, are procedures composed of two algorithm stages [1]. Among these, two main strategies stand out.

- **Cluster-First Route-Second**. The first phase consists of clustering the vertices; then, the second phase determines a feasible route for each cluster.
- **Route-First Cluster-Second**. The first phase consists of generating a large route that visits all the vertices; then, the second phase partitions such route into smaller routes.

These Two-Phase approaches have been widely used in some works for mTSPs. For SmTSP, in [3], a Route-First Cluster-Second strategy is used, and then a GA with intra-route heuristics is used to improve the quality of the routes. Other proposals have used the Cluster-First Route-Second, such as [25], where a variation of the k-means algorithm is used at the clustering phase, then a GA is used to build a route within each cluster. In fact, most works have used variations of the k-means algorithm for the clustering phase for mTSPs [14, 20, 19]. An interesting point of [19] is that the authors used a parallel approach to improve the running times. Regarding MmTSP the situation is similar, variations of the k-means clustering have been used in [21, 17]. It is worth noting that, for the second phase, most authors have used GAs, ant-based algorithms, and hybridizations between them.

It is important to remark that, although there are many proposals for the mTSPs, just a few focuses on the specific variant of this paper (DFmTSP). In fact, in the literature on mTSPs, many works study the SmTSP, but they refer to it just as the mTSP. On the contrary, there are also a few papers where the DFmTSP is studied but referred to as the MmTSP. This is deeply clarified in [8]. This paper studies the DFmTSP by considering two key points: depots do not exist in the problem statement, and there are bounding constraints. As far as we know, a few papers approach this specific variant. Among the last heuristic/metaheuristic proposals that consider these specific constraints, Zhou et al. [26] proposes a Partheno Genetic Algorithm (PGA) that considers lower-bound constraints. Also, [11] proposes a metaheuristic combining an Ant Colony and a PGA. This algorithm is called AC-PGA and considers both lower-bound and upper-bound constraints. According to the presented experimentation in [11], AC-PGA outperforms other proposals in terms of finding better quality solutions.

Regarding the objective functions for the mTSPs, two popular objective functions were initially considered in [4]. The first one is called *minsum* mTSP, where the objective is to minimize the sum of the cost of the salespersons' paths. The second objective function is known as the *minmax* mTSP, which consists of minimizing the longest path among the salespersons. However, the first one has become the most popular in the literature.

3 A Two-Phase Constructive Heuristic

This section introduces a Two-Phase constructive heuristic for the DFmTSP with upper-bound constraint. The proposal is based on the capacitated vertex k-center problem (CVKCP).

Along with k-means, k-center problems are natural clustering methods. In particular, the capacitated version imposes load-balance by considering an upper bound on the number of *clients* each center can attend. The vertex k-center problem (VKCP) has been used in the literature to design efficient constructive heuristics for the DFmTSP [18]. Nevertheless, it can not be used for the DFmTSP with bounding constraints since the VKCP does not restrict the maximum number of vertices that can be assigned to each center. As far as we know, the capacitated version has not been used as a clustering strategy for mTSPs in the literature. Thus, this work explores the advantages of using the capacitated vertex k-center problem as a clustering technique for the DFmTSP with bounding constraints. One of the main advantages, is that the CVKCP can create clusters with a maximum number of assigned vertices to each center, this characteristic is useful for the DFmTSP to limit the number of assigned vertices in each path, which is equivalent to the upper-bound constraint for mTSPs. Algorithm 1 shows the pseudocode of the Two-Phase proposed heuristic. The notation for this algorithm is the following:

- m is the number of salespersons.
- $p = \{p_1, p_2, \cdots, p_m\}$ is a solution for the DFmTSP composed by m salespersons paths.
- p_i is a salesperson path (a sequence of vertices) of a solution p.
- k is the number of centers for the CVKCP.
- C is the set of centers $C \subseteq V(G)$.

- P_C is the assignment function $P_C : V(G) \setminus C \to C$.
- U is the maximum number of vertices each salesperson can visit (upper-bound constraint).
- P_{c_j} is the subset of the assignment that contains only tuples of the form (u, c_j).
- $dom\left(P_{c_j}\right)$ is the set of vertices assigned to be covered by center c_j.

Algorithm 1: Two-Phase Constructive Heuristic.

```
Input: A weighted graph $G = (V, E, w)$, and two positive integers $m$
       and $U$
Output: A set of salespersons tours $p = \{p_1, p_2, \cdots, p_m\}$
1 $p \leftarrow \emptyset$
2 $k \leftarrow m$
3 $(C, P_C) \leftarrow kCenterClustering(G, k, U)$
4 foreach $c_i \in C$ do
5     $X \leftarrow dom\left(P_{c_i}\right) \cup \{c_i\}$
6     $p_i \leftarrow TSPRouting(G[X])$
7     $p \leftarrow p \cup \{p_i\}$
8 end
9 return $p$
```

3.1 Clustering Phase

As mentioned before, the k-center algorithms have been used for clustering purposes. Thus, in this section we propose the usage of a heuristic that has proven to be effective in approaching the CVKCP. The used algorithm is known in the literature as the One-Hop Farthest-First heuristic (OHFF) [7]. This heuristic is based on an exact algorithm for the CVCKP [6], and exploits a relationship between the CVKCP and other combinatorial optimization problem known as the Minimum Capacitated Dominating Set (MCDS). Besides, in [6] is stated the CVKCP can be solved through a series of MCDS problems. The formal relationship is described in the Theorem 1 whose detailed proof can be consulted at [6]. It is known that the CVKCP and the MCDS are both NP-hard. Thus, in a general overview, the OHFF tries to solve the CVKCP by greedily trying to solve MCDS subproblems through a binary search. One of the main features of the OHFF is that parallel computing can be used to improve the running times. Algorithm 2 shows the pseudocode of the OHFF.

Theorem 1. *The minimum capacitated dominating set (MCDS) over the bottleneck graph $G_{OPT} = (V, E_{OPT})$ is the optimal solution to the CVKCP over the original input graph $G = (V, E, w)$, where OPT is the value of the optimal solution to the latter problem.*

Algorithm 2: One-Hop Farthest-First (OHFF) for the CVKCP [7].

```
Input: A complete weighted graph $G = (V, E, w)$, two positive integers
       $k$ and $U$, and a non-decreasing list of the $m$ edge weights of $G$,
       i.e., $w(e_1), w(e_2), ..., w(e_m)$, where $w(e_i) \leq w(e_{i+1})$
Output: A set of vertices $C \subseteq V$, such that $|C| = k$,
        and an assignment $P_C : V \setminus C \rightarrow C$
1  $high \leftarrow m$
2  $low \leftarrow 1$
3  $(C, P_C) \leftarrow (\emptyset, \emptyset)$
4  while $low \leq high$ do
5      $mid \leftarrow \lfloor (high + low)/2 \rfloor$
6      $(C', P_{C'}) \leftarrow GreedyMCDS\,(G, k, w(e_{mid}), U)$
7      if $r(C', P_{C'}) \leq r(C, P_C)$ then
8          $(C, P_C) \leftarrow (C', P_{C'})$
9      end
10     if $r(C, P_C) \leq w(e_{mid})$ then
11         $high \leftarrow mid - 1$
12     else
13         $low \leftarrow mid + 1$
14     end
15 end
16 while $|C| < k$ do
17     $v \leftarrow \arg\max \{d\,(u, P_C\,(u)) : u \in V \setminus C\}$
18     $P_C \leftarrow P_C \setminus \{(v, P_C\,(v))\}$
19     $C \leftarrow C \cup \{v\}$
20 end
21 foreach $c_i \in C$ do
22     $X \leftarrow dom\,(P_{c_i}) \cup \{c_i\}$
23     $c_j \leftarrow \arg\min \{\max\{d(u, v) : v \in X\} : u \in X\}$
24     $P_C \leftarrow P_C \setminus P_{c_i}$
25     $P_{c_j} \leftarrow \{(v, c_j) : X \setminus \{c_j\}\}$
26     $P_C \leftarrow P_C \cup P_{c_j}$
27 end
28 return $(C, P_C)$
```

3.2 Routing Phase

In the routing literature, many algorithms have been proposed for the classical Traveling Salesperson Problem (TSP) [5]. Among these proposals, there are some exact algorithms that guarantee to find the optimal solution. However, since TSP is $\mathcal{NP}$-hard, such algorithms may have an important limitation. Other proposals are approximation algorithms that do not guarantee finding an optimal solution but a solution inside a ratio of the optimal one. Likewise, heuristics do not guarantee optimality but are fast and very effective in finding near-optimal solutions. Metaheuristics are more elaborated procedures that use exploration and exploitation components to escape from local optimals during search. Lin–Kernighan heuristic (LKH) is one of the best heuristics for the TSP [10]. It is a local search algorithm that improves an input tour (Hamilton cycle) by exploring its neighborhood. Every time a shorter tour is found, the process is repeated until no better tour can be found. For this specific heuristic, a neighborhood is defined by considering the number of edges that are in one tour but not the other. For the routing phase of our proposal, we use the Lin-Kernighan algorithm since it has proven to be effective for TSP instances with thousand of vertices. Besides, an efficient implementation is provided in `http://webhotel4.ruc.dk/~keld/research/LKH/`.

4 Computational Experimentation and Analysis

We performed an empirical evaluation of the proposal over some instances of the TSPLIB dataset. For comparison purposes, we implemented the AC-PGA metaheuristic [11], which is one of the best metaheuristics for this specific variant of the problem. For the experimentation, we used the m values in the set $\{5, 10\}$, and for the upper bound U we used the values in $\{\lceil n/m \rceil, \lceil n/m \times 1.1 \rceil\}$. We refer to the value $\lceil n/m \rceil$ as a 0% of imbalance whereas $\lceil n/m \times 1.1 \rceil$ as a 10% of imbalance. The algorithms were implemented in the C++ programming language. All the experiments were carried out on a platform with Intel Core i9-13900, 64 GB RAM, under an OS Ubuntu 22.04.4 LTS 64-bit with a GCC 11.4.0 compiler. The version of the LKH is 2.0.10. All datasets and the implementation of the Two-Phase constructive heuristic can be consulted in `https://gitlab.com/alex.ca/DFmTSP-TP`. The parameter setting of the LKH used in our proposal, and the configuration of the AC-PGA are shown in Tables 1 and 2.

Tables 3 and 4 show the obtained results for 0% and 10% of imbalance of the three tested algorithms, where OHFF+ is the OHFF heuristic but executed $|V(G)|$ times with a different initial chosen vertex. The objective function is minsum. For the **AC-PGA** column, f_{best} is the objective value of the best-found solution of 30 independent runnings, whereas f_μ is the average, σ is the standard deviation, and $t(s)$ is the average running time in seconds. For the **OHFF/LKH** column, f_{best} is the objective value of the best-found solution of performing 30 independent runnings of the OHFF for the CVKCP and then running the LKH in the routing phase, f_μ is the average of the 30 runnings, σ is the standard deviation, and $t(s)$ is the average of the sum of both running times, the clustering phase and the routing phase. Due to the nature of the OHFF algorithm, some solutions for the CVKCP may include clusters with only one vertex.

In these cases, a salesperson path can not be constructed. Then, such solutions were ignored. For the **OHFF+/LKH**, the columns are the same as the previous but with the difference that OHFF+ was used. Note that the standard deviation of the latter is always 0 because the OHFF+ algorithm is deterministic. From these tables, we observe that when the percentage of imbalance is 0% (Table 3), the AC-PGA was capable of finding some of the best solutions.

Table 1. Parameter setting of the LKH [10].

Parameter	Value
RUNS	1
TIME_LIMIT	0.1s
MOVE_TYPE	5
PATCHING_C	3
PATCHING_A	2

Table 2. Parameter setting of the AC-PGA metaheuristic [11].

Parameter	Value
Population size	100
AC-PGA iterations	100
ACO iterations	100
ρ	0.1
α	2
β	8
γ	0.5

Nevertheless, an important remark is that the running times of AC-PGA are much higher than the proposals that use the OHFF as the clustering phase. For the case where 10% of imbalance is allowed (Table 4), most of the best solutions were found by the proposals that use the OHFF and the LKH. Thus, we conclude that, at least for the tested instances, the proposed Two-Phase heuristic is capable of finding better solutions when imbalance is allowed among the paths. Furthermore, the running times of the Two-Phase heuristic are many orders of magnitude lower than those of the AC-PGA metaheuristic.

Fig. 1 shows the printed solutions by the tested algorithms over the instance kroA200 with a 10% of imbalance. From this figure, we can observe that the best-found solutions by the AC-PGA contain some overlaps between the paths of the salespersons. On the contrary, solutions computed by the Two-Phase heuristic proposals have fewer overlaps, and the paths are better refined due to the LKH.

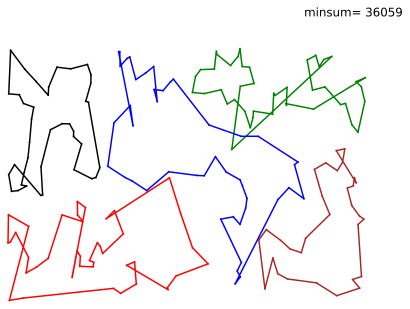
Fig. 2 and Fig. 3 show the convergence of the AC-PGA metaheuristic over the instance kroA200 with 0% and 10% of imbalance respectively. In these figures, the dotted lines represent the objective values of the solutions computed by the Two-Phase heuristic proposals. It is important to note that the Two-Phase heuristic proposals do not have generations since they are constructive heuristics. Nevertheless, they are shown in the figures for contrast purposes. In these figures, it can be observed that the proposals can find good quality solutions compared to the AC-PGA. However, due to the nature of AC-PGA metaheuristic, its exploration and exploitation components could cause the algorithm to escape from local optima, which could give the possibility that during the generations, the solution found may eventually be better than those found by the Two-Phase heuristic proposals, such is the case of the Fig. 2(b).

Table 3. Results over some instances of the TSPLIB dataset with 0% of imbalance. The best-found solutions are bold.

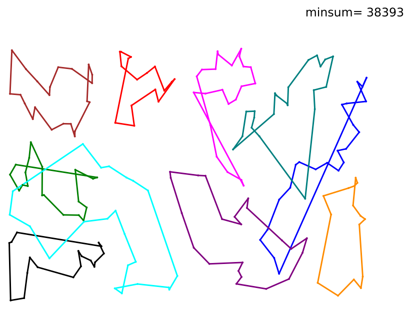
Instance	n	m	U	AC-PGA				OHFF/LKH				OHFF+/LKH			
				f_{best}	f_{μ}	σ	$t(s)$	f_{best}	f_{μ}	σ	$t(s)$	f_{best}	f_{μ}	σ	$t(s)$
kroA100	100	5	20	25016	25823	485.28	87	**23554**	27864	1640.57	0.0023	**23554**	23554	0	0.035
		10	10	**26593**	27371	487.34	86	29984	36203	2975.18	0.0033	30895	30895	0	0.058
kroB100	100	5	20	**24742**	26100	619.73	87	28223	30112	748.05	0.0022	24821	24821	0	0.036
		10	10	**27332**	28421	622.52	87	33756	40930	3799.44	0.0031	29802	29802	0	0.057
kroA150	150	5	30	30872	32510	618.19	199	**30513**	35621	2386.67	0.0027	31577	31577	0	0.098
		10	15	**33056**	34869	620.26	195	36779	44823	2763.87	0.0029	40862	40862	0	0.139
kroB150	150	5	30	30730	32320	565.49	199	29894	33882	3070.23	0.0023	**29224**	29224	0	0.095
		10	15	**33826**	35106	643.54	194	34829	46986	5240.63	0.0027	35186	35186	0	0.129
kroA200	200	5	40	35503	36994	722.84	355	34618	37381	1257.45	0.0020	**32284**	32284	0	0.206
		10	20	38968	40348	605.29	347	40126	46134	3567.72	0.0027	**35367**	35367	0	0.318
kroB200	200	5	40	34831	36393	577.19	355	**33191**	36795	1159.41	0.0022	34059	34059	0	0.218
		10	20	36222	38667	961.79	347	**35539**	47045	4400.87	0.0027	36715	36715	0	0.272
pr226	226	5	46	105262	107910	1586.42	452	**102824**	118669	9376.52	0.0016	107812	107812	0	0.167
		10	23	**110059**	114034	1961.42	442	126819	152624	17007.07	0.0026	125730	125730	0	0.233
pr264	264	5	53	**58210**	59999	895.17	617	60286	60864	443.44	0.0019	60552	60552	0	0.297
		10	27	53252	54709	831.13	608	**50868**	58201	7412.33	0.0024	51762	51762	0	0.301
pr299	299	5	60	58254	59787	687.26	791	55176	56996	2044.10	0.0022	**54629**	54629	0	0.351
		10	30	**62728**	64485	855.80	782	65223	72845	5111.33	0.0030	67588	67588	0	0.548
pr439	439	5	88	132677	136023	1630.25	1706	**117113**	121844	3746.26	0.0044	118061	118061	0	1.124
		10	44	**140475**	143925	1412.78	1682	140638	167431	13625.24	0.0076	145451	145451	0	1.803

Table 4. Results over some instances of the TSPLIB dataset with 10% of imbalance. The best-found solutions are bold.

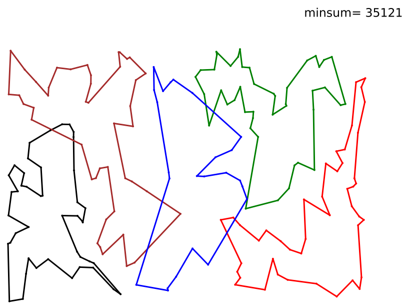
Instance	n	m	U	AC-PGA				OHFF/LKH				OHFF+/LKH			
				f_{best}	f_{μ}	σ	$t(s)$	f_{best}	f_{μ}	σ	$t(s)$	f_{best}	f_{μ}	σ	$t(s)$
kroA100	100	5	22	25116	25896	425.99	87	**24418**	28151	1706.43	0.0021	24499	24499	0	0.031
		10	11	26582	27463	413.00	87	**25605**	31282	3347.16	0.0032	26745	26745	0	0.051
kroB100	100	5	22	25651	26300	384.15	87	25651	29820	1790.54	0.0021	**25647**	25647	0	0.035
		10	11	**26262**	27985	901.65	87	27186	29427	1405.30	0.0029	27560	27560	0	0.052
kroA150	150	5	33	31301	32756	551.65	199	31393	34347	2015.50	0.0023	**29732**	29732	0	0.092
		10	17	33740	35252	667.75	196	32192	36397	3994.71	0.0026	**31820**	31820	0	0.110
kroB150	150	5	33	31251	32432	548.34	199	**28861**	32195	2963.20	0.0024	29601	29601	0	0.082
		10	17	32713	34371	646.82	196	**30496**	33561	1539.51	0.0031	31463	31463	0	0.111
kroA200	200	5	44	36059	37283	491.31	355	35121	38331	2317.41	0.0020	**32343**	32343	0	0.201
		10	22	38393	40014	609.98	348	36279	38180	2100.52	0.0027	**34599**	34599	0	0.246
kroB200	200	5	44	35456	36605	553.96	355	**33495**	37481	2284.02	0.0022	34716	34716	0	0.205
		10	22	37361	39150	655.16	349	**35087**	41324	4751.43	0.0028	35946	35946	0	0.244
pr226	226	5	50	**98788**	104878	2516.27	449	100707	111289	6742.86	0.0015	107997	107997	0	0.144
		10	25	**96528**	107709	3589.47	441	105342	129225	10807.50	0.0017	127776	127776	0	0.213
pr264	264	5	59	**59181**	61050	622.76	617	60789	60798	45.77	0.0020	60789	60789	0	0.263
		10	30	52405	54504	948.33	608	50480	55964	5127.32	0.0021	**50144**	50144	0	0.314
pr299	299	5	66	59125	60544	622.18	792	**52861**	58107	2706.32	0.0018	53936	53936	0	0.324
		10	33	62671	65695	1022.61	792	**56627**	62990	5636.68	0.0025	58514	58514	0	0.454
pr439	439	5	97	131110	135665	1700.13	1817	**115286**	119869	2922.18	0.0050	125807	125807	0	1.081
		10	49	139017	144026	1815.28	1682	**127688**	139649	6790.23	0.0042	137703	137703	0	1.413



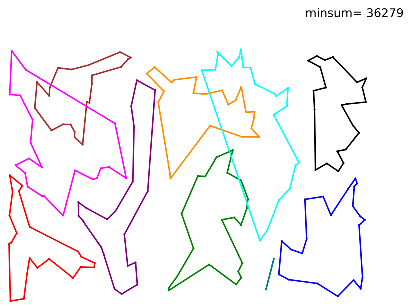
(a) AC-PGA, $m = 5$



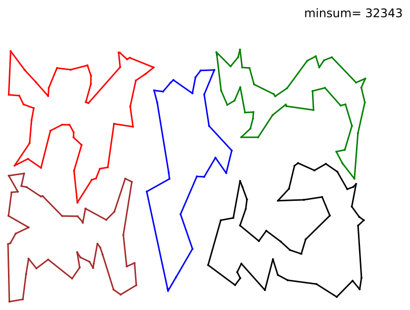
(b) AC-PGA, $m = 10$



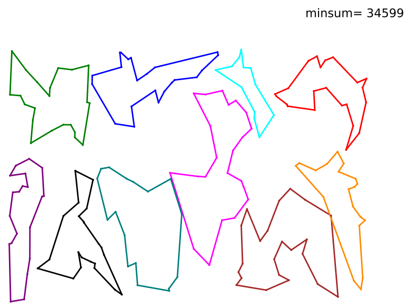
(c) OHFF/LKH, $m = 5$



(d) OHFF/LKH, $m = 10$

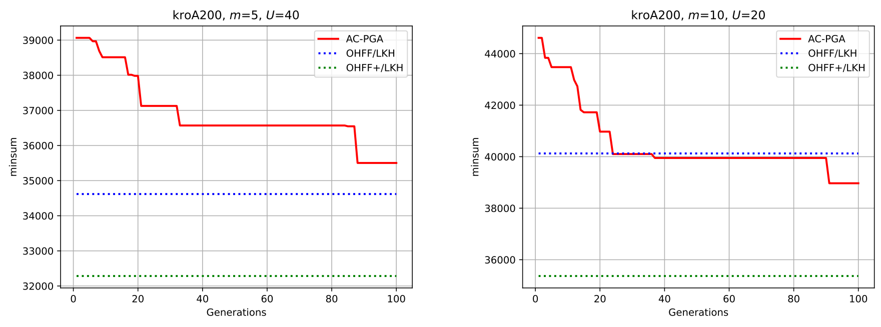


(e) OHFF+/LKH, $m = 5$



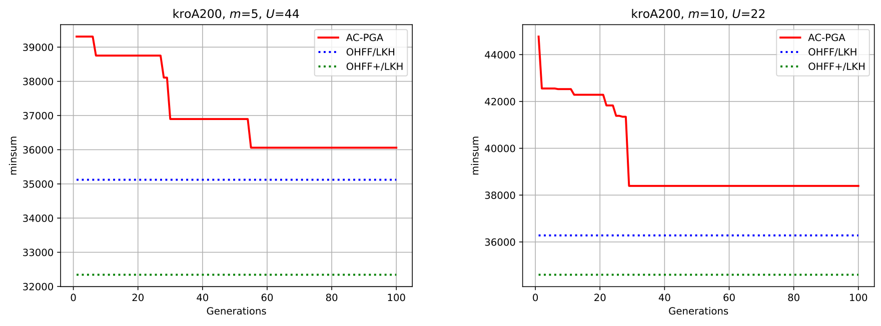
(f) OHFF+/LKH, $m = 10$

Fig. 1. Printed solutions of the kroA200 instance with 10% of imbalance.



(a) (b)

Fig. 2. Convergence plot of AC-PGA over instance kroA200 with 0% of imbalance. OHFF/LKH and OHFF+/LKH are included for comparative purposes.



(a) (b)

Fig. 3. Convergence plot of AC-PGA over instance kroA200 with 10% of imbalance. OHFF/LKH and OHFF+/LKH are included for comparative purposes.

5 Conclusions and Future Work

In this paper, a Two-Phase constructive heuristic for the Depot-Free Multiple Traveling Salesperson Problem (DFmTSP) was proposed. The main feature of our proposal is that a state-of-the-art heuristic for the capacitated vertex k-center problem (CVKCP) is used in the clustering phase, which is known as the One-Hop Farthest-First (OHFF). For the routing phase, a state-of-the-art heuristic called Lin-Kernighan (LKH) was used. The obtained results show that the proposed Two-Phase heuristic was able to find feasible and good-quality solutions in comparison with a state-of-the-art metaheuristic that employs elaborated exploration and exploitation mechanisms. Besides, reported running times support that the proposed Two-Phase heuristic is a good choice when practical running times matter. Some future work directions may arise from this research. For instance, working with dynamic clusters may be of interest. This can be performed through an evolutionary or metaheuristic that employs components to handle partitions of the set of vertices, then applying intensification methods over the partitions such as the LKH to find better quality solutions iteratively. This approach may lead to finding better solutions in comparison to working with static clusters/partitions, just as we worked in this research.

Acknowledgments. We acknowledge Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT), Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), and Tecnológico Nacional de México / ITS de Purísima del Rincón, for providing the necessary resources for the development of this research.

References

1. Anbuudayasankar, S. P., Ganesh, K., Mohapatra, S.: Models for Practical Routing Problems in Logistics. Springer Cham, 1 edn. (2014)
2. Barbato, M., Gouveia, L.: The Hamiltonian p-median problem: Polyhedral results and branch-and-cut algorithms. European Journal of Operational Research, vol. 316, no. 2, pp. 473–487 (2024) doi: 10.1016/j.ejor.2024.02.032
3. Bolaños, R. I., Toro O, E. M., Granada E, M.: A population-based algorithm for the multi travelling salesman problem. International Journal of Industrial Engineering Computations, vol. 7, no. 2, pp. 245–256 (2016) doi: 10.5267/j.ijiec.2015.10.005
4. Carter, A. E., Ragsdale, C. T.: A new approach to solving the multiple traveling salesperson problem using genetic algorithms. European Journal of Operational Research, vol. 175, no. 1, pp. 246–257 (2006) doi: 10.1016/j.ejor.2005.04.027
5. Cook, W. J.: In Pursuit of the Traveling Salesman. Princeton University Press, New Jersey, 1 edn. (2012)
6. Cornejo-Acosta, J. A., García-Díaz, J., Menchaca-Méndez, R., Menchaca-Méndez, R.: Solving the Capacitated Vertex K-Center Problem through the Minimum Capacitated Dominating Set Problem. Mathematics, vol. 8, no. 9, pp. 1551 (2020) doi: 10.3390/math8091551

7. Cornejo-Acosta, J. A., García-Díaz, J., Pérez-Sansalvador, J. C., Ríos-Mercado, R. Z., Pomares-Hernández, S. E.: A Constructive Heuristic for the Uniform Capacitated Vertex k-center Problem. ACM J. Exp. Algorithmics, vol. 28 (2023) doi: `10.1145/3604911`
8. Cornejo-Acosta, J. A., García-Díaz, J., Pérez-Sansalvador, J. C., Segura, C.: Compact Integer Programs for Depot-Free Multiple Traveling Salesperson Problems. Mathematics, vol. 11, no. 13 (2023) doi: `10.3390/math11133014`
9. Gavish, B.: A NOTE ON "THE FORMULATION OF THE M-SALESMAN TRAVELING SALESMAN PROBLEM". Management Science, vol. 22, no. 6, pp. 704–705 (1976) doi: `10.1287/mnsc.22.6.704`
10. Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research, vol. 126, no. 1, pp. 106–130 (2000) doi: `10.1016/S0377-2217(99)00284-2`
11. Jiang, C., Wan, Z., Peng, Z.: A new efficient hybrid algorithm for large scale multiple traveling salesman problems. Expert Systems with Applications, vol. 139, pp. 112867 (2020) doi: `10.1016/j.eswa.2019.112867`
12. Kara, I., Bektas, T.: Integer linear programming formulations of multiple salesman problems and its variations. European Journal of Operational Research, vol. 174, no. 3, pp. 1449–1458 (2006) doi: `10.1016/j.ejor.2005.03.008`
13. Karabulut, K., Öztop, H., Kandiller, L., Tasgetiren, M. F.: Modeling and optimization of multiple traveling salesmen problems: An evolution strategy approach. Computers & Operations Research, vol. 129, pp. 105192 (2021) doi: `10.1016/j.cor.2020.105192`
14. Latah, M.: Solving Multiple TSP Problem by K-Means and Crossover based Modified ACO Algorithm. International Journal of Engineering Research & Technology (IJERT), vol. 5, pp. 430–434 (2016)
15. Li, Y., Soleimani, H., Zohal, M.: An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives. Journal of Cleaner Production, vol. 227, pp. 1161 – 1172 (2019) doi: `10.1016/j.jclepro.2019.03.185`
16. Miller, C. E., Tucker, A. W., Zemlin, R. A.: Integer Programming Formulation of Traveling Salesman Problems. J. ACM, vol. 7, no. 4, pp. 326–329 (1960) doi: `10.1145/321043.321046`
17. Mohammed, Y., Al-Khateeb, B.: A Novel Metaheuristic Algorithm for Multiple Traveling Salesman Problem. Journal of Advanced Research in Dynamical and Control Systems, vol. 10, pp. 2113–2122 (2018)
18. Na, B.: HEURISTIC APPROACH FOR THE NO-DEPOT K-TRAVELING SALESMEN PROBLEM WITH A MINMAX OBJECTIVE. Master's thesis, Texas A&M University (2006), `https://hdl.handle.net/1969.1/5825`
19. Othman, A., Haimoudi, E. k., Mouhssine, R., Ezziyyani, M.: An Effective Parallel Approach to Solve Multiple Traveling Salesmen Problem. In: Ezziyyani, M. (ed) Advanced Intelligent Systems for Sustainable Development (AI2SD'2018). pp. 647–664. Springer International Publishing, Cham (2019) doi: `10.1007/978-3-030-11928-7_58`
20. Sathya, N., Muthukumaravel, A.: Two Phase Hybrid AI-Heuristics for Multiple Travelling Salesman Problem. International Journal of Applied Engineering Research, vol. 12, pp. 12659–12664 (2017)
21. Steven, A., Hertono, G. F., Handari, B. D.: Implementation of clustered ant colony optimization in solving fixed destination multiple depot multiple traveling salesman problem. In: 2017 1st International Conference on Informatics and Computational Sciences (ICICoS). pp. 137–140 (2017) doi: `10.1109/ICICOS.2017.8276351`

22. Svestka, J. A., Huckfeldt, V. E.: COMPUTATIONAL EXPERIENCE WITH AN M-SALESMAN TRAVELING SALESMAN ALGORITHM. Management Science, vol. 19, no. 7, pp. 790–799 (1973) doi: 10.1287/mnsc.19.7.790
23. Wang, M., Ma, T., Li, G., Zhai, X., Qiao, S.: Ant Colony Optimization With an Improved Pheromone Model for Solving MTSP With Capacity and Time Window Constraint. IEEE Access, vol. 8, pp. 106872–106879 (2020) doi: 10.1109/ACCESS.2020.3000501
24. Wang, Z., Fang, X., Li, H., Jin, H.: An Improved Partheno-Genetic Algorithm With Reproduction Mechanism for the Multiple Traveling Salesperson Problem. IEEE Access, vol. 8, pp. 102607–102615 (2020) doi: 10.1109/ACCESS.2020.2998539
25. Xu, X., Yuan, H., Liptrott, M., Trovati, M.: Two phase heuristic algorithm for the multiple-travelling salesman problem. Soft Computing, vol. 22, no. 19, pp. 6567–6581 (2018) doi: 10.1007/s00500-017-2705-5
26. Zhou, H., Song, M., Pedrycz, W.: A comparative study of improved GA and PSO in solving multiple traveling salesmen problem. Applied Soft Computing, vol. 64, pp. 564–580 (2018) doi: 10.1016/j.asoc.2017.12.031

Electronic edition
Available online: http://www.rcs.cic.ipn.mx