SoftwareX 12 (2020) 100601

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

PackageCargo: A decision support tool for the container loading problem with stability

Juan Martínez-Franco, Edgar Céspedes-Sabogal, David Álvarez-Martínez *

University of Los Andes, School of Engineering, Cra 1 N° 18A - 12 Bogotá D.C, Colombia

ARTICLE INFO

Article history:
Received 5 May 2020
Received in revised form 29 September 2020
Accepted 30 September 2020

Keywords:
Optimization
Cargo stability
Physics engine
Unity

ABSTRACT

This article presents PackageCargo. A modular open-source application developed using the Unity game engine to calculate, visualize, and save efficient packing patterns to instances of the Container Loading Problem (CLP). The packing patterns are obtained through approximate optimization algorithms (metaheuristics). Additionally, the proposed tool allows us to estimate cargo stability metrics through the implementation of mathematical models and verify the results of said models using a simulation environment built with the PhysX library. The goal of this application was to create a usable decision support system suitable for industrial purposes as well as a platform for academic research. It is offering a modifiable framework that can adapt to the necessities of its users, saving them software development time while continuing to extend PackageCargo through community contributions. The resulting application was compared with commercial software solutions. Furthermore, each module was tested using the most successful approaches found in literature as benchmarks. The packing module was compared against the top-performing algorithm published to date, obtaining similar results in similar computational times. The simulation module for cargo stability was benchmarked against high-performance simulation software, validating its accuracy and performance. Accordingly, PackageCargo was found to have a competitive feature set useful in both academic and commercial settings. As future work, it is proposed to combine the different modules to solve more sophisticated variants of the CLP, like the container loading problem constrained to weight distribution profiles.

Code metadata

Current code version	v1.0.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_198
Code Ocean compute capsule	
Legal Code License	GPL 3.0
Code versioning system used	git
Software code languages, tools, and services used	C++, C#, MySQL, Unity, PhysX
Compilation requirements, operating environments & dependencies	Unity Editor 5.6.6 or higher
If available Link to developer documentation/manual	
Support email for questions	jc.martinez10@uniandes.edu.co

Software metadata

Current software version	1.0.1
Permanent link to executables of this version	For example: https://github.com/jcmartinez10/PackageCargo/releases/tag/v1.0.1-alpha
Legal Software License	GPL-3.0
Computing platforms/Operating Systems	Microsoft Windows, distributed/web based
Installation requirements & dependencies	Unity Editor 5.6.6 or higher
If available, link to user manual - if formally published include a reference to the publication in the reference list	https://github.com/jcmartinez10/PackageCargo/blob/master/README.md
Support email for questions	jc.martinez10@uniandes.edu.co

* Corresponding author.
E-mail address: d.alvarezm@uniandes.edu.co (D. Álvarez-Martínez).

https://doi.org/10.1016/j.softx.2020.100601

1. Motivation and significance

Cargo transportation presents significant logistical challenges that, although frequently explored in academic settings, have been scarcely addressed by freight companies and customers. One of such challenges is related to the arrangement of cargo within containers, a task that is formally defined as the Container Loading Problem (CLP). The CLP has a broad spectrum of applicability as well as mathematical characteristics that make it attractive to researchers and industry alike, but novel strategies to approach it have not been widely deployed in real-world situations. This might stem from software tools used to solve the challenges being proprietary or licensed and, therefore, unavailable for smaller actors or use in academic settings. In response to the described situation, we introduce an open-source tool concerned with solutions to a variant of the CLP that considers the physical behavior of cargo.

There are multiple applications focused on generating basic solutions for the CLP. Features in these applications range from simple geometric optimization to network connectivity (Cubemaster®), center-of-gravity related constraints (LoadCargo®) and orientation and load-bearing constraints (EasyCargo®). To the best of our knowledge, no software package provides all of the functionality we sought to include in this application, particularly regarding cargo stability and software extensibility. Among explored software options, we could not find a free alternative with feature parity to the commercial solutions available, nor applications that made their source code available.

To address this, a new application developed using the game development suite Unity® (C #), using a modular design that includes the calculation of solutions to the CLP called packing patterns, the visualization of said solutions and a simulation environment to estimate the cargo stability of the cargo. The open-source nature of the proposed tool (PackageCargo) allows researchers as well as the general public to integrate their developments, whether they are oriented towards simulation or optimization without the need to implement an entire framework. In this way, works focused on simulation are coupled to a default optimization algorithm that allows finding efficient packaging patterns. When compared with commercial solutions, the proposed application has a competitive feature set.

1.1. The container loading problem

The Container Loading Problem consists of packing rectangular boxes into a rectangular space or container, making sure that boxes do not overlap and that volume utilization is maximized [1]. This problem is of great importance among cutting and packing problems, being cataloged as an NP-Hard problem [2]. This classification leads to the CLP typically being approached using approximate optimization algorithms.

Some factors that add to the complexity of the CLP are the practical constraints associated with variants of the problem. Bortfeldt [3] compiled a set of functional constraints applied to the CLP, and among them, this work pays significant attention to cargo stability. Several authors have considered cargo stability as the most essential constraint [4–6] because it influenced the integrity of cargo (customer satisfaction) and the safety of operators performing loading and unloading operations. In addition to cargo stability, PackageCargo considers the constraints of box orientation, load-bearing strength, weight limits, full support positioning, and split loading at multiple destinations (multidrop constraint).

1.2. Cargo stability

Cargo stability refers to the balance of the cargo during transportation. In essence, freight is considered to be stable if its geometric configuration and physical integrity are preserved during loading and unloading operations as well as during accelerations when being transported. Two metrics used to evaluate cargo stability are the number of fallen boxes (NFB) and the number of boxes within the damaged boundary curve (NB_DBC). The former refers to the number of boxes permanently displaced along the vertical axis of the container between the initial and the final states of container movement. At the same time, the later is the number of boxes that suffer either a drastic change in velocity or an excessive acceleration for a given moment. The acceptable acceleration and velocity delta values are based on the tests as described in [7], used to determine the fragility of items. Typically, stability metrics are obtained from running a dynamic simulation either on dedicated, high precision simulation software or real-time simulation middleware known as physics engines.

2. Software description

PackageCargo is an application that produces and visualize packing patterns for cargo instances. Cargo instances are structured lists of items, with predefined sizes, orientation constraints, and load-bearing constraints that are to be loaded into a container of a predetermined size. Cargo definitions, as well as generated patterns, can be visualized, saved, and retrieved (See Fig. 1). Packing patterns can also be subjected to a simulation that returns stability metrics using different techniques.

2.1. Software architecture

Fig. 2 illustrates the architecture of the program. The architecture consists of four modules: visualization, composed of a renderer and a user interface that includes simulation and optimization results as well as performance indicators; simulation module, that tests the cargo stability of the loading pattern; optimization, that allows the inclusion of different algorithms for the generation of packing patterns; and a database that manages registered data along with data generated by the optimization and simulation modules.

Visualization of the packages and the container in a 3D environment is achieved using Unity's rendering engine, supported by a wide variety of rendering APIs [8]. Unity also includes an extensive UI library that allowed for the creation of the interface for the tool (see Fig. 3).

Dynamic simulations are performed either with the physics engine provided by Unity (Nvidia's PhysX) or with an algorithm based on a simplified mechanical model of the cargo [9], with the possibility to incorporate a different simulation module defined by the user. PhysX presents two critical limitations: it produces stochastic results that may vary over multiple executions, and it prioritizes performance over accuracy [10]. The accuracy of the physics engine was tested through a benchmark using the high precision simulation environment found in Autodesk Inventor [9].

The packing pattern generator uses the GRASP algorithm developed in [11] to generate efficient packing patterns. Still, this task can be performed by a multitude of algorithms that can be linked to the application directly through memory or system files.

The optimization module has been associated with the GRASP algorithm included by default [11]. This algorithm presents a competitive performance in the 3500 classic instances of the CLP [12], as shown in Table 1. Table 1 illustrates a fair comparison between the GRASP approach and the best-in-class published

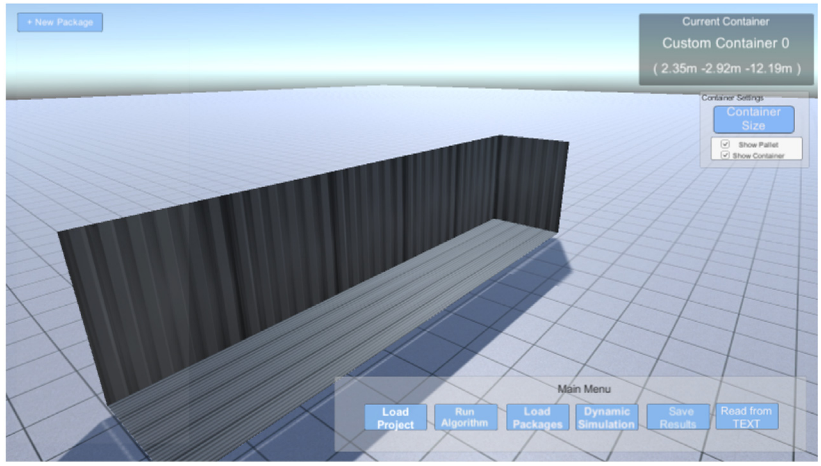

Fig. 1. Application interface. The different menu options are related to each of the modules that make up the tool. The container and cargo configurations are updated and visualized in real time. Some visualization options are available, including toggleable visibility for containers and/or pallets.

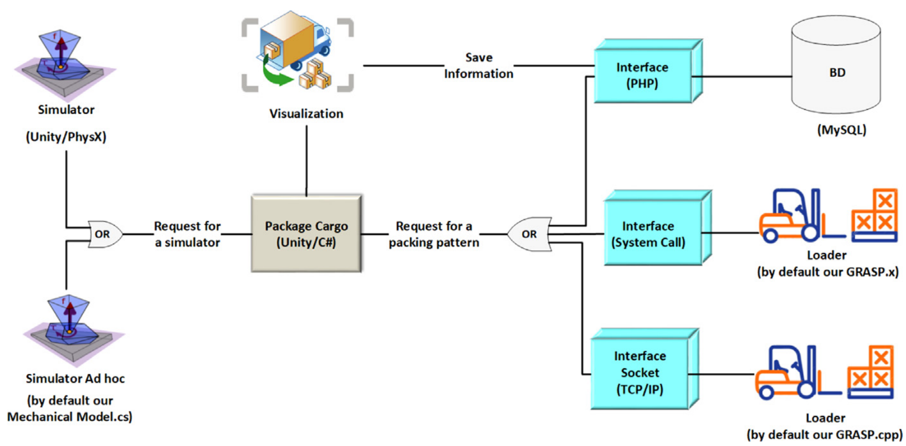

Fig. 2. Architecture of PackageCargo. Different protocols connect the application client to an external database, a server run packing algorithm or a local instance of that type of algorithm. Function calls request for a simulation result from either the default physics engine (PhysX) or a custom solution. Visualization is managed by internal Unity settings on the client side.

method for solving the container loading problem considering constraints relating to box orientation, weight limits, full package support, load-bearing limits, and multiple cargo destinations. A group of 100 instances is conformed for each number of customers (1,2,5,10,50) and family of problems (BR1–BR7), and the average volume utilization is reported for each group. The algorithm achieves competitive volume utilization percentages on CLP instances with the mentioned constraints within a reasonable computing time.

Nevertheless, it is possible to use other loading algorithms by a socket connected to a server or by directly calling executable programs so that users can compare them and generate reports on the performance indicators relating to stability. For the streaming of data through the socket, a connection by TCP/IP is established in a client–server architecture, meaning that external loading algorithms must be defined as servers that receive requests from the client application. In the case of executable files, the user must build their algorithm as executable supported by their target platform (Windows, macOS, or Linux).

The data storage module manages all information, whether it be registered by the user or generated by the other modules. Results are recorded on a database (MySQL®) located on a centralized server, which any user can access by request. This module performs calls to log, update, and delete information related to user-created instances. It has a PHP interface that modifies the database by SQL commands. The database stores cargo information, container size, packing patterns, and (optionally) cargo stability metrics.

3. Example usage

PackageCargo can produce and visualize packing patterns for CLP instances, taking into account some of the most relevant constraints published in the literature. In PackageCargo, the 1500 classic test instances of the literature are stored by default [12], and the user can define a new instance and also store it. When an algorithm is executed to solve a test case immediately, its result is stored in the cloud. This way, the user can decide to load an existing experiment (a test instance or a solution and its performance metrics) or start a new one.

Starting a new experiment involves defining cargo and container properties, as shown in Fig. 3. When a user creates an instance and executes the optimization algorithm for it, PackageCargo sends the structured data of the instance to the "API-Loader" executable that is run on the server-side, and then receives the solution data.

Afterward, the user has the option to save the data (doing so will store both said instance and the solution generated in a remote database), calculate several performance metrics, or even manually modify the solution.

Table 1
GRASP performance (measured as volume utilization percentage and computational time) against the best-in-class algorithm [11].

Customers	GRASP								Christensen & Rousoe
	BR1	BR2	BR3	BR4	BR5	BR6	BR7	Average	
1	92,47	92,75	92,76	92,22	91,68	91,01	89,62	91,79	89,07
2	91,77	90,76	89,47	88,56	87,78	86,69	85,51	88,65	85,96
5	88,56	86,02	83,15	81,36	80,14	78,30	76,12	81,95	78,52
10	85,28	81,43	77,82	76,14	74,47	72,45	70,22	76,83	72,64
50	79,32	73,96	68,98	66,87	64,85	63,25	60,61	68,26	64,45
Average	87,48	84,98	82,44	81,03	79,78	78,34	76,42	81,50	78,13
Time (s)	30,29	34,01	37,93	41,98	57,18	63,35	69,94	47,81	60,00

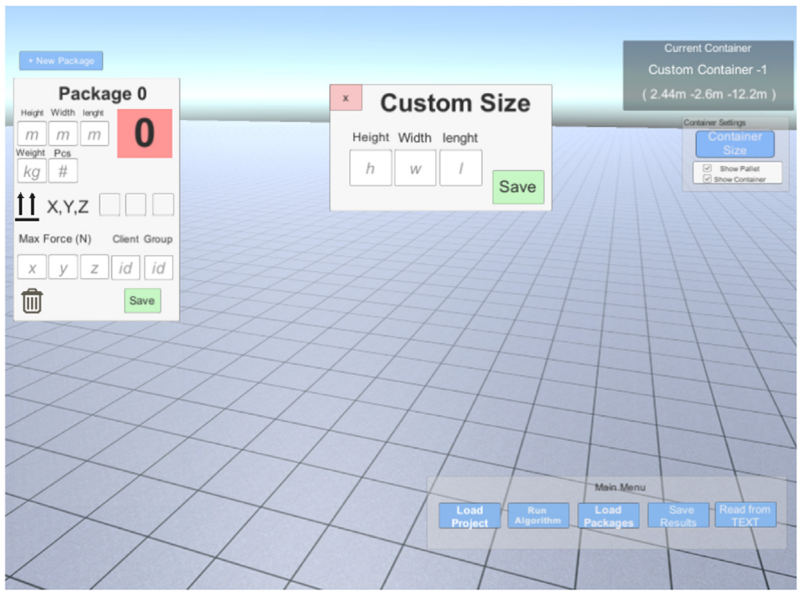

Fig. 3. Definition of a new CLP instance in the application. The graphical interface allows the user to define cargo varieties, dimensions, constraints and quantities. This information is converted to an input file for packing pattern generating algorithms.

4. Extensibility

The software architecture allows for different modules of the program to be replaced by user-defined ones. For example, the dynamic simulations of the engine can be exchanged for a simple mechanical model that returns the same simulation metrics. The function call for dynamic simulations can even refer to run physics engines externally.

As an example of an alternative simulation environment, we include a motion mechanics based algorithm that estimates cargo stability from the packing configuration solution. The algorithm is run instead of the standard dynamic simulation pipeline by modifying a simple function call in the simulation GUI.

Similarly, the optimization module that produces solutions to instances of the container loading problem can be switched by a user-defined algorithm. The hook to a new algorithm can lead to both a local executable file or to an external socket solution. The packaged solution is based on an implementation of a GRASP algorithm that is run on the client-side.

5. Impact

PackageCargo provides significant contributions on two fronts: commercial applications and research aid. In the former, our software tool achieves functionality comparable to that of specialized industrial applications such as EasyCargo [13] and LoadCargo [14]. Together with a sophisticated algorithmic development of optimization and simulation, it positions itself as an alternative to paid software. But at the same time, it encourages stakeholders (industry actors and researches) to complement it with the features they consider necessary.

The open-source nature of PackageCargo makes the software readily accessible for contributions seeking to enhance its capabilities beyond the scope of this work. It also makes the offered optimization and simulation tools available to the scientific and academic communities to allow the evaluation of simulation models and optimization algorithms.

This software has already been used to test a novel algorithm capable of predicting cargo stability indicators on a very short. The simulation module was replaced with a simple call to a C# method that made use of the cargo definition parameters to produce results in much shorter times compared to dynamic simulations. Enhancements were included in the software.

6. Conclusions

We introduced PackageCargo, a modular software tool capable of obtaining solutions to instances of the Container Loading Problem. PackageCargo allows for the definition of cargo properties, container size, and shipping constraints, as well as the algorithm, used to generate packing patterns for the defined instances. The software has already been used in research involving the cargo stability constraint and is thought to provide a viable alternative for customers using existing commercial solutions. The tool presents a modular architecture allowing both academic and industry sectors to easily extend or modify the application to better suit their needs. Additionally, instances and algorithms can be saved and accessed remotely using the network features present in the application.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This project was carried out as part of a SIC project, financed by the Colombian Ministry of Science, Technology and Innovation.

References

[1] Wäscher G, Haußner H, Schumann H. An improved typology of cutting and packing problems. E J Oper Res 2007;183(3):1109–30.
[2] Pisinger D. Heuristics for the container loading problem. European J Oper Res 2002;141(2):382–92.
[3] Bortfeldt A, Wäscher G. Constraints in container loading – A state-of-the-art review. European J Oper Res 2013;229(1):1–20.
[4] Bischoff EE. Stability aspects of pallet loading. OR Specktrum 1991;13(4):189–97.
[5] Moura A, Oliveira JF. A GRASP approach to the container-loading problem. IEEE Intell Syst 2005;20(4):50–7.
[6] Parreño F, Alvarez R, Oliveira J, Tamarit J. A maximal-space algorithm for the container loading problem. Inform J Compt 2008;20(3):412–22.
[7] ASTM International. Standard test methods for mechanical-shock fragility of products. 2010.
[8] Unity Technologies. Unity es él software lder a nivel mundial en la industria de los juegos. 2017, Available Online http://www.unity.com/. Accesed (02/02/2017).
[9] Martinez-Franco J, Alvarez-Martinez D. PhysX As a Middleware for Dynamic Simulations in the Container Loading Problem in Proceedings of the 2018 Winter Simulation Conference. 2018.
[10] PhysX, is a scalable multi-platform game physics solution. 2016, Available Online https://developer.nvidia.com/physx-sdk. Accesed (20/12/2016).
[11] Alvarez-Martínez D, Alvarez R, Parreño F. A grasp algorithm for the container loading problem with multi-drop constraints. Pesq Oper 2015;35:1–24.
[12] Bischoff EE, Ratcliff MSW. Issues in the development of approaches to container loading. Omega 1995;23:377–90.
[13] EasyCargo. Easycargo is truck and container loading software. 2017, Available Online http://www.easycargo3d.com/. Accesed(16/04/2017).
[14] LoadCargo. Container and Truck Loading Software. 2017, Available Online https://www.loadcargo.in/. Accesed(16/04/2017).