COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

PRH-ANP: PROGRAMA INTERDEPARTAMENTAL DE TECNOLOGIAS DIGITAIS PARA O SETOR DE PETRÓLEO E GÁS

EMSO_OLCA:

Ferramenta para Cálculo de Análise de Ciclo de Vida integrada ao EMSO

Bolsista: Simone de Carvalho Miyoshi, D.Sc.

Supervisor: Prof. Argimiro Secchi, D.Sc.

Janeiro/2023

prh Programa de Recursos Humanos da ANP
GESTÃO: Finep

anp Agência Nacional do Petróleo, Gás Natural e Biocombustíveis

Finep INOVAÇÃO E PESQUISA
EMPRESA PÚBLICA DO MCTI

Sumário

1. EMSO-OLCA

A ferramenta EMSO_OLCA foi desenvolvida em C++ para integrar o software de cálculo de análise de ciclo de vida OpenLCA ao Ambiente para Modelagem, Simulação e Otimização de Processos – EMSO. Foi desenvolvida no Projeto de Pós-doutorado da pesquisadora Simone Miyoshi. O desenvolvimento dessa ferramenta só foi possível através do financiamento pela bolsa de pós-doutorado do Programa de Recursos Humanos da Agência Nacional do Petróleo, Gás e Biocombustíveis PRH 4.1.

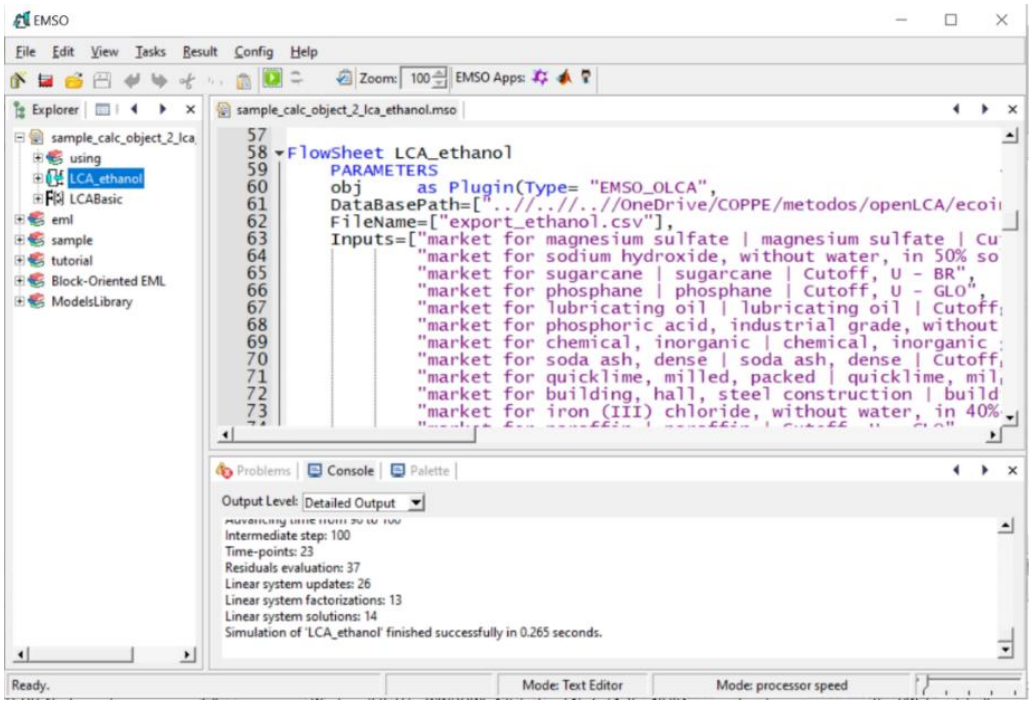
O uso da ferramenta é gratuito para uso acadêmico, contudo, pedimos citar o trabalho:

Miyoshi, S.C.; Matias, B.S.; Secchi, A.R. Life Cycle Assessment Tool Integrated to Process Monitoring, Design and Control Environment. *In:* 11th Word Congress of Chemical Engineering, 2023, Buenos Aires.

A integração da análise de ciclo de vida ao ambiente para modelagem e simulação de processos é particularmente interessante pois permite além do monitoramento on-line de emissões, o design de equipamentos baseados nos princípios da bioeconomia, avaliação tecno-econômica ambiental de processos, otimização de processos e de controle baseados em métricas ambientais.

O EMSO é um ambiente para modelagem, simulação e otimização de processos, gratuito para uso acadêmico (SOARES e SECCHI, 2003). Possui rapidez no tempo de Simulação quando comparado com os simuladores comerciais, o que é propício para aplicações em tempo real. Possui Integração com OPC, Python, Matlab, Scilab / OPC / Excel, LibreOffice, além de facilidade de integração com sistemas industriais em tempo real (SOARES e SECCHI, 2003), conforme mostra na Figura 1**.**

Figura 1 – Ambiente para Modelagem, Simulação e Otimização de Processos



Fonte: de autoria própria

O OpenLCA, apresentado na Figura 2, é um programa de computador livre e gratuito para cálculo de análise de ciclo de vida (OPENLCA, 2022), contudo nem todas as bases de dados de inventários são gratuitas e muitas requerem licenças específicas. As bases de dados de inventários podem ser adquiridas no sitio https://nexus.openlca.org/. Atualmente, a COPPE/UFRJ adquiriu licenças das bases de dados EcoInvent 3.8 e Agri-footprint 5.

Figura 2– Programa de Computador de Avaliação de Ciclo de Vida - OpenLCA

Fonte: de autoria própria

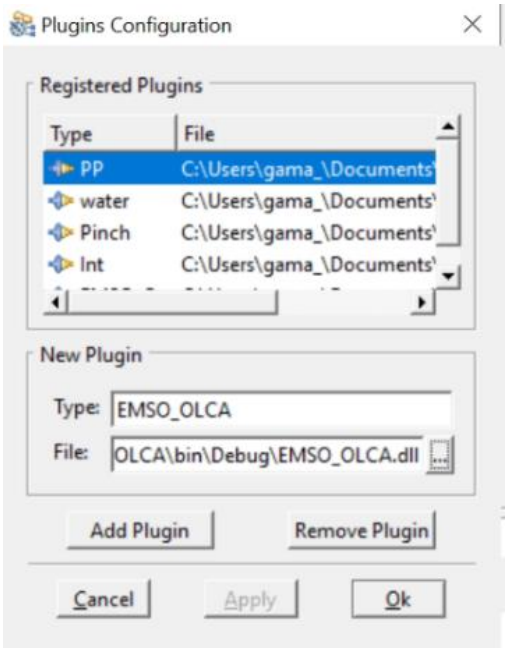
Passo a passo para utilização da ferramenta:

1) Instalação da Ferramenta EMSO_OLCA;
2) *Download* das Metodologias de Avaliação de Ciclo de Vida no EMSO-OLCA;
3) Cálculo das ACVs das entradas no OpenLCA e exportação dos resultados;
4) Configuração dos parâmetros do Plugin no EMSO.

1.1. Instalação da Ferramenta EMSO_OLCA

A biblioteca EMSO-OLCA está disponível atualmente para ambiente Windows sendo testada para o EMSO versões 0.10.10 e 0.10.11. Para instalação, o caminho do arquivo EMSO_OLCA.dll deve ser informado no EMSO através do *Menu* > *Config* > *Plugins*, como na Figura 2. Em *Type* digitar: EMSO_OLCA e em File indicar o caminho da EMSO_OLCA.dll. Como sugestão colocar na pasta EMSO\interface.

Figura 3– Instalação da Ferramenta EMSO_OLCA



Fonte: de autoria própria

1.2. Metodologias de Avaliação de Ciclo de Vida no EMSO_OLCA

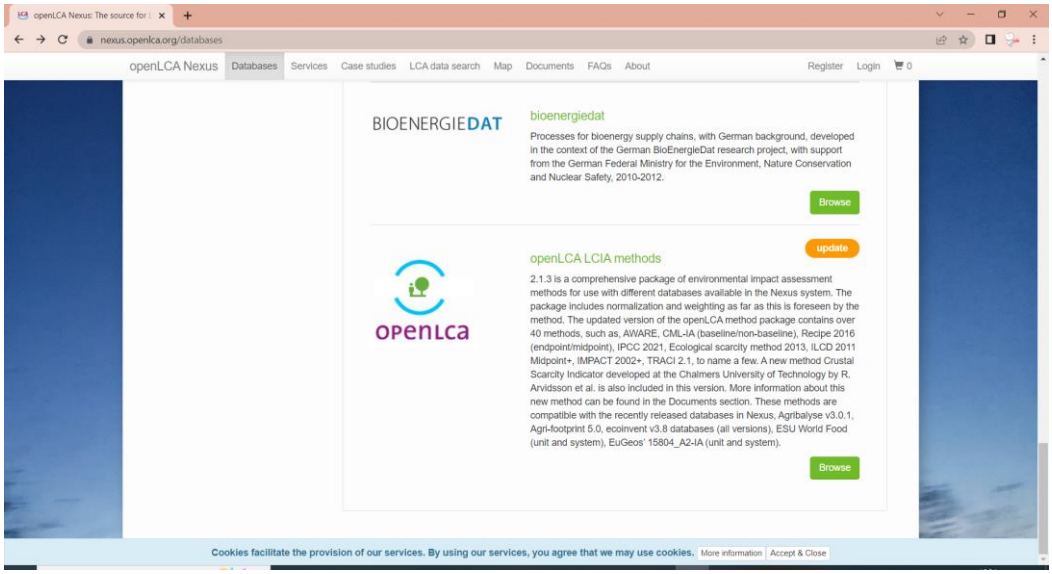
As Metodologias de Avaliação de Ciclo de Vida são um conjunto de categorias de impacto que quantificam o impacto de cada emissão. Por exemplo: Para a categoria de impacto *Climate Change* no método CML baseline, o impacto do metano corresponde a 28 CO_2 equivalentes. Estão disponíveis 99 metodologias de impacto do OpenLCA (versão 2.2.1) e do EcoInvent (versão 3.8) com um total de 1479 categorias de impacto.

1.2.1. *Download* das Metodologias de Avaliação de Ciclo de Vida no EMSO_OLCA

A base de dados de Metodologias de Avaliação de Ciclo de Vida do EMSO-OLCA está disponível no sitio do NEXUS-OPENLCA (nexus.openlca.org/databases). O arquivo com as metodologias pode ser baixado gratuitamente, conforme mostra a Figura 4. O formato de integração com o EMSO-OLCA é o JSON-LD.

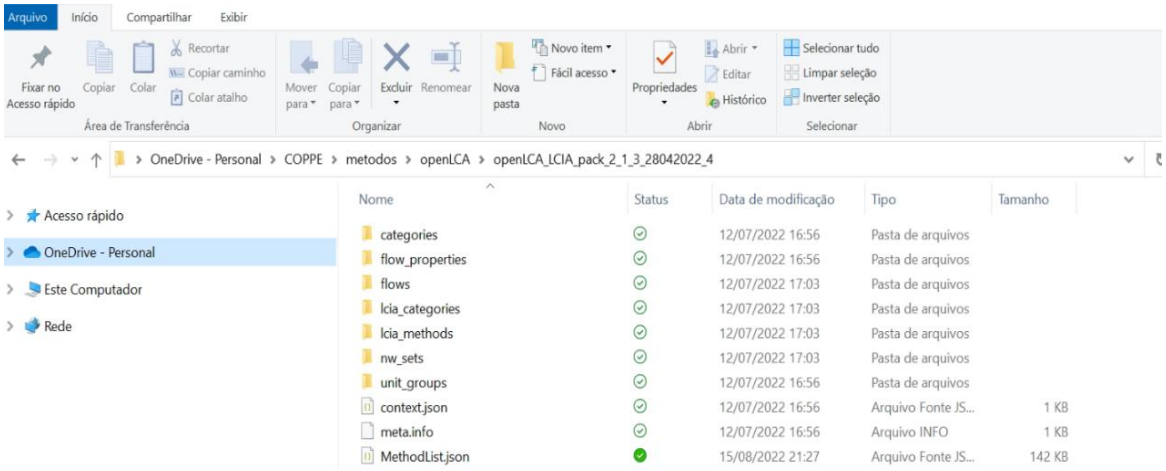
A versão do arquivo das metodologias do openLCA devem ser compatíveis com a versão do banco de dados de inventário (LCI). Cada usuário do EMSO_OLCA deve baixar a pasta com os arquivos de formato JSON individualmente. A pasta deverá ser descompactada em uma pasta específica e acessível ao usuário, sem precisar de senha de administrador, conforme a Figura 5. O endereço da pasta deverá ser informado nas configurações do objeto EMSO_OLCA no parâmetro ***DataBasePath.***

Figura 4 - Metodologias de Avaliação de Ciclo de Vida disponíveis no OpenLCA



Fonte: de autoria própria

Figura 5 – Caminho dos métodos do OpenLCA



Fonte: de autoria própria

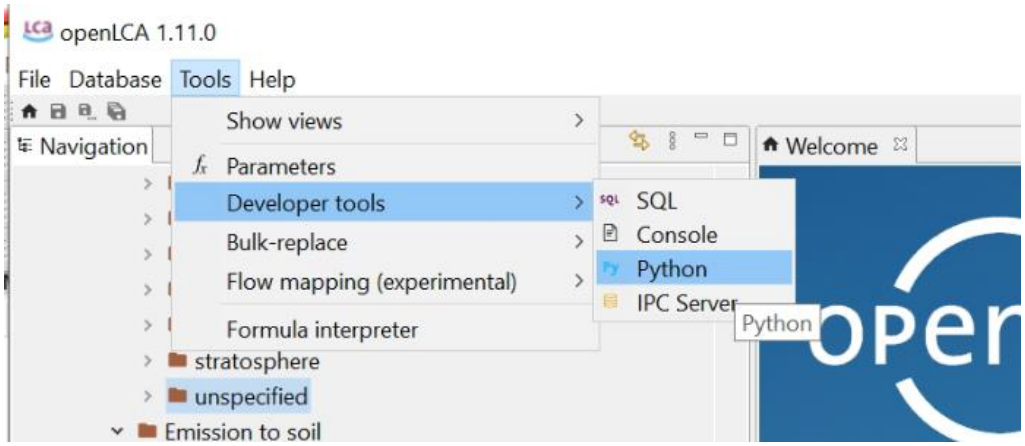
Essa pasta não será distribuída juntamente com o EMSO-OLCA por questões de Licenças de Distribuição, contudo a licença dos métodos pelo OpenLCA é gratuita. Sem essa pasta não é possível realizar a Análise de Ciclo de Vida no EMSO-OLCA. Futuros desenvolvimentos a se considerar podem vir a utilizar uma biblioteca de métodos própria. Contudo, o uso das metodologias diretamente no sitio NEXUS-OpenLCA permite utilizar uma grande variedade de metodologias já testadas e atualizadas constantemente no OpenLCA.

O caminho da pasta da base de dados, o nome da metodologia e os nomes das categorias de impacto devem ser informados no EMSO com como parâmetros ***DataBasePath***, ***MethodName*** e ***Characterization_Factor***, respectivamente.

1.3. Exportação do arquivo com as entradas

As entradas do EMSO-OLCA são importadas diretamente do OpenLCA e o seu impacto é calculado através do Menu TOOLS> Developer Tools > Python, conforme a Figura 6.

Figura 6 - Ferramenta do Desenvolvedor Python no OpenLCA

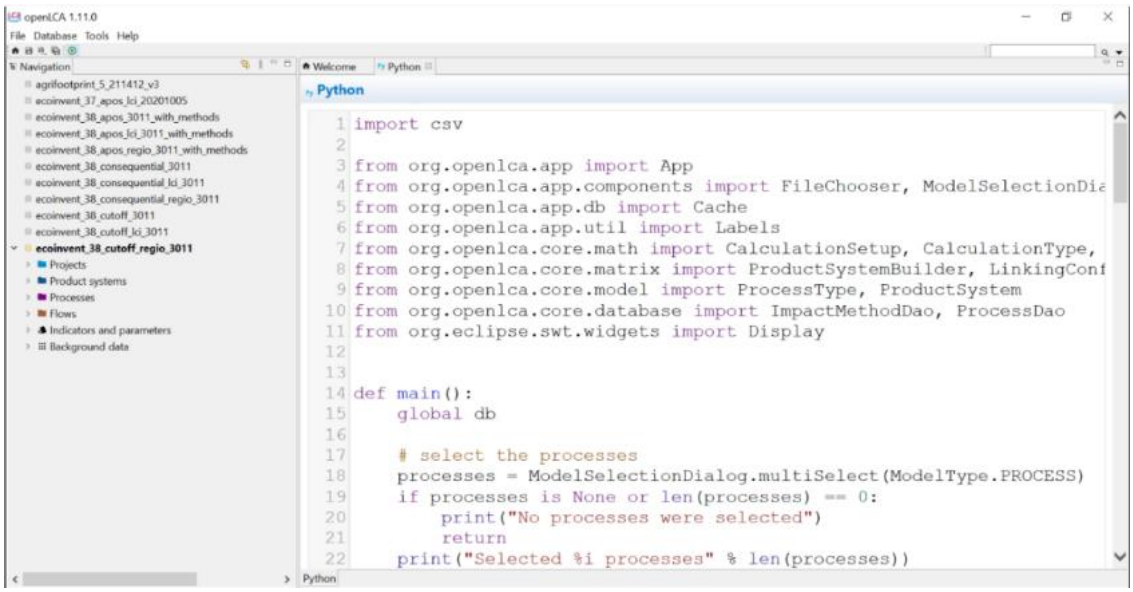


Fonte: de autoria própria

Para exportar as entradas em um arquivo .csv, deve-se:

- executar o código em Python (disponível no anexo), conforme mostra a Figura 7;
- selecionar as entradas a serem consideradas;
- selecionar o método de Avaliação do Ciclo de Vida;
- indicar o local para salvar o arquivo .csv com os resultados dos impactos das entradas.

Figura 7 - Interface de Python no OpenLCA



Fonte: de autoria própria

Os nomes dos processos de entrada bem como o nome do arquivo de exportação dos resultados da ACV são configurados no EMSO como: ***Inputs*** e ***InputFileName***.

1.3. Fluxos Elementares

Os fluxos elementares são definidos como recursos naturais, emissões ou resíduos. Esses devem ser configurados conforme o tipo (emissão para o ar, água, solo ou recurso ou resíduo ou ainda emissão imaterial) e conforme o local, como constam as definições do OpenLCA. As opções disponíveis são apresentadas na Tabela 1.

Tabela 1 – Fluxos Elementares no OpenLCA/ EMSO-OLCA (em inglês)

ElementaryFlowPath1	ElementaryFlowPath2
Emission to air	high population density
Emission to air	low population density
Emission to air	low population density, long-term
Emission to air	lower stratosphere + upper troposphere
Emission to air	stratosphere
Emission to air	unspecified
Emission to soil	agricultural
Emission to soil	forestry
Emission to soil	industrial
Emission to soil	unspecified
Emission to soil	urban, non industrial
Emission to water	fossil-
Emission to water	fresh water
Emission to water	fresh water, long-term
Emission to water	ground water
Emission to water	ground water, long-term
Emission to water	lake
Emission to water	ocean
Emission to water	river
Emission to water	river, long term
Emission to water	surface water
Emission to water	unspecified
Immaterial emission	unspecified
Resource	biotic
Resource	in air
Resource	in ground
Resource	in water
Resource	land
Resource	unspecified
Waste	ecopoints 97, CH
Waste	unspecified

A lista dos nomes dos fluxos elementares, do tipo de fluxo elementar e do local do fluxo elementar é configurado pelo usuário no EMSO em **ElementaryFlows**, **ElementaryFlowPath1 e ElementaryFlowPath2** respectivamente.

1.4. Conversão de Unidades no EMSO-OLCA

As unidades das entradas do EMSO-OLCA são definidas conforme o tipo de variável: variável de entrada e variável de saída. As unidades de entradas e saídas e seus fatores de conversão estão configurados pela sua característica padrão definida na base de dados do OpenLCA. Essas unidades estão definidas no arquivo EMSO_OLCA_units.csv. O nome com caminho completo do EMSO_OLCA_units.csv deve ser informado nas configurações do usuário. De forma geral, utilizar as entradas, saídas e fluxos elementares em kg.

1.5. Configuração dos Usuários no EMSO

Os usuários devem inserir um parâmetro no EMSO do tipo ***Plugin*** com a seguinte sintaxe:

```
PARAMETERS

obj    as Plugin (Type= "EMSO_OLCA",
DataBasePath=["C:/Users/usuario/Documents/EMSO_OLCA/methods_database"],
MethodName= ["MethodName"],
ImpactCategory=["ImpactCategory1", " ImpactCategory2"],
InputFileName=["C:/Users/usuario/Documents/EMSO_OLCA/export_input.csv"],
Inputs=["name of input1", "name of input 2"],
OutputName=["name of the output 1"],
OutputUnit=["kg"],
ElementaryFlows=["Emission1","Emission2","ResourceName1","ResourceName"],
ElementaryFlowPath1=["Emission to air", "Emission to air", "Resource","Resource"],
ElementaryFlowPath2=["low population density","low population density","in water",
"land"],
UnitFileName=["C:/Users/usuario/Documents/EMSO_OLCA/EMSO_OLCA_units.csv"],
MethodologyType=["attributional"], # ou "consequential"
AllocationType=["mass"]); # ou "energy" ou "economic"
```

1.6. Modelo Básico no EMSO

O modelo básico, no EMSO do EMSO_OLCA, já apresenta algumas funções implementadas de forma a facilitar o uso pelo usuário, conforme anexo B. O modelo implementado também apresentado na Figura 8.

Figura 8 – Modelo Básico do LCA no EMSO

```
Model LCABasic
    PARAMETERS
outer NoComps as Integer;
outer obj as Plugin(Type="EMSO_OLCA");

    ni as Integer (Brief="Number of Inputs", Default=2);
    ncf as Integer (Brief="Number of Characterization Factors", Default=3);
    no as Integer (Brief= "Number of Outputs", Default=2);
    ne as Integer (Brief= "Number of ElementaryFlows (emissions+resources)", Default=2);
    nc as Integer (Brief= "Number of Consequential Input", Default=0);
    alloc_par(no) as Real (Default=1, Lower=1e-5);


    VARIABLES

    cf(ncf, ni) as Real;
    ef(ncf, ne) as Real;
    r (ncf, no) as Real;
    p (ncf, ni+ne+nc) as Real;

    input_values(ni) as Real (Default=1);
    emission_values(ne) as Real(Default=1);
    output_values(no) as Real (Default=1, Lower=1e-5);

    EQUATIONS

    [ef(1,:),ef(2,:),ef(3,:),ef(4,:),ef(5,:),ef(6,:),ef(7,:),ef(8,:),ef(9,:),ef(10,:),ef(11,:)] = obj.EmissionFactor();

    [cf(1,:),cf(2,:),cf(3,:),cf(4,:),cf(5,:),cf(6,:),cf(7,:),cf(8,:),cf(9,:),cf(10,:),cf(11,:)]  = obj.CharacterizationFactor();

    # keep input values and emissions values in kg
    [r(1,:), r(2,:), r(3,:), r(4,:), r(5,:), r(6,:), r(7,:), r(8,:), r(9,:), r(10,:), r(11,:)] =
                                    obj.LcaCalc(input_values, emission_values, output_values, alloc_par);

    [p(1,:), p(2,:), p(3,:), p(4,:), p(5,:), p(6,:), p(7,:), p(8,:), p(9,:), p(10,:), p(11,:)] =
                                    obj.LcaPercentual(input_values, emission_values, output_values, alloc_par);

end
```

Fonte: de autoria própria

As funções disponíveis estão apresentadas na Tabela 2.

Tabela 2 – Funções disponíveis no EMSO_OLCA

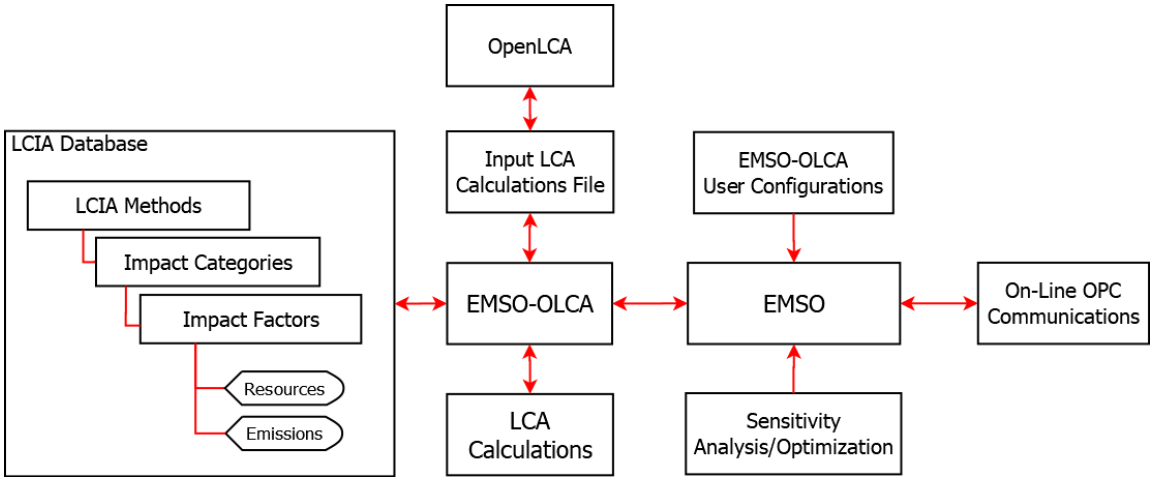
Função	Descrição
EmissionFactor	Fator de Emissão de cada emissão de acordo com os fatores de caracterização indicados.
CharacterizationFator	Fator de Emissão das entradas de acordo com os fatores de caracterização indicados.
LcaCalc	Cálculo do LCA
LcaPercentual	Cálculo do impacto de cada entrada e fluxo elementar no cálculo do LCA

Em seguida, apresenta-se o funcionamento da ferramenta.

1.7. Funcionamento da Ferramenta

O fluxo de informações no EMSO-OLCA é descrito na Figura 9. O EMSO lê as configurações do usuário e informa ao EMSO-OLCA. O EMSO-OLCA consulta a Base de Dados do LCIA e obtém os fatores de impacto de acordo com os fatores de caracterização definidos. O EMSO-OLCA também procura um arquivo de cálculo de LCA das entradas que é criado pelo aplicativo Python no OpenLCA. Em seguida, o EMSO-OLCA realiza os cálculos de LCA adequados conforme a metodologia especificada pelo usuário no EMSO.

Figura 9 - Fluxo de informações EMSO_OLCA



Fonte: de autoria própria

1.8. Cálculo da ACV

O valor calculado do impacto da categoria c $(LCIA_c)$ é obtido a partir do produto do Fator de Impacto $IF_{c,i}$ do componente i para a categoria c e a quantidade do componente i emitida (LCI_i), como na equação 19 (Jolliet et al., 2010):

$$LCIA_c = \sum_i IF_{c,i} \times LCI_i \tag{1}$$

Para os fluxos não elementares o fator de impacto é calculado através do próprio software OpenLCA. No caso dos fluxos elementares, o fator de impacto é obtido diretamente da lista de fatores de cada categoria de impacto.

Nos métodos de alocação atribucionais, o cálculo do fator de alocação é dado pela Equação 20 (Guinée, 1995):

$$f_i = \frac{par_i \cdot m_i}{\sum_j (par_j \cdot m_j)} \tag{2}$$

Para alocação mássica, $par_i = 1$. Para alocação energética $par_i = conteúdo\ energético_i\ (MJ/kg)$. E para alocação econômica, $par_i = preço_i\ (\$/kg)$.

1.9. Teste da Análise de Ciclo de Vida do Etanol de Cana-de-açúcar

Foi realizado um teste da ferramenta EMSO_OLCA utilizando-se o Inventário da base de dados EcoInvent 3.8 para um estudo de caso de produção de Etanol para unidades autônomas. Os resultados estimados utilizando-se metodologia CML baseline são apresentados na Tabela 10.

Tabela 10 – Resultados do Teste para Análise de Ciclo de Vida do Etanol (Unidades Autônomas, BR) (*em inglês*)

Categoria de Impacto	Resultado OpenLCA	Resultado EMSO	Erro Percentual
Photochemical oxidation	0,00169	0,00169359	0,2124%
Human toxicity	0,25019	0,250187	0,0012%
Abiotic depletion	4,29159E-06	4,29155E-06	0,0009%
Eutrophication	0,00741	0,00740821	0,0242%
Abiotic depletion (fossil fuels)	3,28619	3,28619	0,0000%
Marine aquatic ecotoxicity	257,25495	257,255	0,0000%
Ozone layer depletion (ODP)	1,9625E-08	1,96253E-08	0,0000%
Terrestrial ecotoxicity	0,00109	0,0010923	0,2110%
Acidification	0,01395	0,0139457	0,0308%
Fresh water aquatic ecotox.	0,13882	0,138824	0,0029%
Global warming (GWP100a)	0,06515	0,0651403	0,0149%

A Tabela 10 mostra que o EMSO_OLCA é capaz de reproduzir os cálculos do OpenLCA apresentando erros reduzidos em relação ao OpenLCA, que podem ser explicados pela diferença do número de casas decimais do resultado. O erro médio percentual (MAPE) é dado por 0,0453% entre os resultados das categorias avaliadas. Note que o tempo computacional utilizado para o cálculo foi de 0,296 s. Esse tempo é muito reduzido em relação ao tempo de cálculo dos softwares de análise de ciclo de vida comerciais, pois grande parte do tempo computacional é gasto no cálculo prévio do impacto das entradas.

Anexos

A. Código em Python para obtenção do cálculo das entradas no arquivo .csv

```
import csv

from org.openlca.app import App
from org.openlca.app.components import FileChooser, ModelSelectionDialog
from org.openlca.app.db import Cache
from org.openlca.app.util import Labels
from org.openlca.core.math import CalculationSetup, CalculationType, SystemCalculator
from org.openlca.core.matrix import ProductSystemBuilder, LinkingConfig
from org.openlca.core.model import ProcessType, ProductSystem
from org.openlca.core.database import ImpactMethodDao, ProcessDao
from org.eclipse.swt.widgets import Display


def main():
  global db

  # select the processes
  processes = ModelSelectionDialog.multiSelect(ModelType.PROCESS)
  if processes is None or len(processes) == 0:
    print("No processes were selected")
    return
  print("Selected %i processes" % len(processes))

  # select the LCIA method
  method = ModelSelectionDialog.select(ModelType.IMPACT_METHOD)
  if method is None:
    print("No LCIA method was selected")
    return
  indicators = ImpactMethodDao(db).getCategoryDescriptors(method.id)
  print("Selected LCIA method '%s' with %i indicators" %
      (method.name, len(indicators)))
  file_name=method.name +  ".csv";

  # select the CSV file where the results should be written to
  f = FileChooser.forExport('*.csv', 'export.csv')
  #f = FileChooser.forExport('*.csv', file_name)
  if f is None:
    print("No CSV file selected")
    return
  print("Selected CSV file: %s" % f.absolutePath)

  # init the CSV file, run calculations, and write results
  with open(f.getAbsolutePath(), 'wb') as stream:

    # configure the CSV writer
    # see https://docs.python.org/2/library/csv.html
    writer = csv.writer(stream, delimiter=';')

    # write the indicators as column headers
    header = ['Process', 'Type', 'Product', 'Amount', 'Unit']
    for i in indicators:
                header.append(enc('%s (%s)' % (i.name, i.referenceUnit)))
      #header.append(enc('%s (%s)' % (i.name, i.id)))
          #header.append(enc('%s' % (i.name)))
```

```
    writer.writerow(header)

    for d in processes:
      # load the process
      process = ProcessDao(db).getForId(d.id)
      ptype = 'Unit process'
      if process.processType != ProcessType.UNIT_PROCESS:
        ptype = 'LCI result'
      qref = process.quantitativeReference

      # we can only create a product system from a process
      # when it has a quantitative reference flow which is
      # a product output or waste input
      if qref is None:
        print('Cannot calculate %s -> no quant.ref.' % d.name)
        continue

      # prepare the CSV row; we will calculate the results
      # related to 1.0 unit of the reference flow
      row = [
        enc(Labels.getDisplayName(process)),
        ptype,
        enc(Labels.getDisplayName(qref.flow)),
        1.0, # qref.amount,
        enc(Labels.getDisplayName(qref.unit))
      ]

      # build the product system with a configuration
      print('Build product system for: %s' % d.name)
      config = LinkingConfig()
      # set ProcessType.UNIT_PROCESS to prefer unit processes
      config.preferredType = ProcessType.LCI_RESULT
      # provider linking: the other options are IGNORE and PREFER
      config.providerLinking = LinkingConfig.DefaultProviders.ONLY
      builder = ProductSystemBuilder(
        Cache.getMatrixCache(), config)
      system = builder.build(process)
      system.targetAmount = 1.0  # the reference amount

      # run the calculation
      print('Calculate process: %s' % d.name)
      calculator = SystemCalculator(
        Cache.getMatrixCache(), App.getSolver())
      setup = CalculationSetup(
        CalculationType.SIMPLE_CALCULATION, system)
      setup.impactMethod = method
      result = calculator.calculateSimple(setup)

      # write results
      print('Write results for: %s' % d.name)
      for i in indicators:
        value = result.getTotalImpactResult(i)
        row.append(value)
      writer.writerow(row)


def enc(s):
  return unicode(s).encode("utf-8")


if __name__ == "__main__":
  Display.getDefault().asyncExec(main)
```

B. Exemplo no EMSO

```
FlowSheet LCA_ethanol
        PARAMETERS
        obj    as Plugin(Type= "EMSO_OLCA",
        UnitFilePath=["C:/Users/gama_/Documents/EMSO_OLCA/EMSO_OLCA/EMSO_OLCA_units.csv"],
        DataBasePath=["C:/Users/gama_/OneDrive/COPPE/metodos/openLCA/ecoinvent_38_cutoff_3011"],
        InputFileName=["C:/Users/gama_/Documents/EMSO_OLCA/EMSO_OLCA/export_ethanol.csv"],


        Inputs=["market for magnesium sulfate | magnesium sulfate | Cutoff, U - GLO",
                        #"market for magnesium sulfate | magnesium sulfate | Cutoff",

                        "market for sodium hydroxide, without water, in 50% solution state | sodium
hydroxide, without water, in 50% solution state | Cutoff, U - GLO",
                        "market for sugarcane | sugarcane | Cutoff, U - BR",
                        "market for phosphane | phosphane | Cutoff, U - GLO",
                        "market for lubricating oil | lubricating oil | Cutoff, U - RoW",
                        "market for phosphoric acid, industrial grade, without water, in 85% solution state |
phosphoric acid, industrial grade, without water, in 85% solution state | Cutoff, U - GLO",
                        "market for chemical, inorganic | chemical, inorganic | Cutoff, U - GLO",
                        "market for soda ash, dense | soda ash, dense | Cutoff, U - GLO",
                        "market for quicklime, milled, packed | quicklime, milled, packed | Cutoff, U - RoW",
                        "market for building, hall, steel construction | building, hall, steel construction |
Cutoff, U - GLO",
                        "market for iron (III) chloride, without water, in 40% solution state | iron (III) chloride,
without water, in 40% solution state | Cutoff, U - GLO",
                        "market for paraffin | paraffin | Cutoff, U - GLO",
                        "market for dithiocarbamate-compound | dithiocarbamate-compound | Cutoff, U -
GLO",
                        "market for metal working, average for chromium steel product manufacturing | metal
working, average for chromium steel product manufacturing | Cutoff, U - GLO",
                        "market for sodium dichromate | sodium dichromate | Cutoff, U - GLO",
                        "market for sodium sulfate, anhydrite | sodium sulfate, anhydrite | Cutoff, U - RoW",
                        "market for metal working, average for steel product manufacturing | metal working,
average for steel product manufacturing | Cutoff, U - GLO",
                        "market for dimethylamine | dimethylamine | Cutoff, U - RoW",
                        "market for cyclohexane | cyclohexane | Cutoff, U - GLO",
                        "market for concrete, normal | concrete, normal | Cutoff, U - BR",
                        "market for acetic acid, without water, in 98% solution state | acetic acid, without
water, in 98% solution state | Cutoff, U - GLO",
                        "market for sodium metasilicate pentahydrate, 58% active substance, powder |
sodium metasilicate pentahydrate, 58% active substance, powder | Cutoff, U - RoW",
                        "market for pesticide, unspecified | pesticide, unspecified | Cutoff, U - GLO",
                        "market for chemical, organic | chemical, organic | Cutoff, U - GLO",
                        "market for sodium hypochlorite, without water, in 15% solution state | sodium
hypochlorite, without water, in 15% solution state | Cutoff, U - RoW",
                        "market for aluminium sulfate, powder | aluminium sulfate, powder | Cutoff, U -
RoW",
                        "market for formaldehyde | formaldehyde | Cutoff, U - RoW",
                        "market for hydrochloric acid, without water, in 30% solution state | hydrochloric acid,
without water, in 30% solution state | Cutoff, U - RoW",
                        "market for monoethanolamine | monoethanolamine | Cutoff, U - GLO",
                        "market for sulfuric acid | sulfuric acid | Cutoff, U - RoW",
                        "market for calcium chloride | calcium chloride | Cutoff, U - RoW"],
        OutputName=["ethanol, without water, in 95% solution state, from fermentation"],
        OutputUnit=["kg"],
        #OutputUnit=["kg"],

        ElementaryFlows=["Particulates, < 2.5 um",
                                "Particulates, > 2.5 um, and < 10um",
                                "NMVOC, non-methane volatile organic compounds, unspecified origin",
```

```
                                        "Carbon dioxide, biogenic",
                                        "Methane, biogenic",
                                        "Ethanol",
                                        "Dinitrogen monoxide",
                                        "Nitrogen oxides",
                                        "Sulfur oxides",
                                        "Carbon monoxide, land transformation",
                                        "Water, well",
                                        "Transformation, to industrial area, built up",
                                        "Occupation, industrial area, built up",
                                        "Transformation, from arable land, unspecified use"          ],

        ElementaryFlowPath1=["Emission to air",
                                                "Emission to air",
                                                "Emission to air",
                                                "Emission to air",
                                                "Emission to air",
                                                "Emission to air",
                                                "Emission to air",
                                                "Emission to air",
                                                "Emission to air",
                                                "Resource",
                                                "Resource",
                                                "Resource",
                                                "Resource"],

   ElementaryFlowPath2=["low population density",
                                                        "low population density",
                                                        "low population density",
                                                        "low population density",
                                                        "low population density",
                                                        "low population density",
                                                        "low population density",
                                                        "unspecified",
                                                        "low population density",
                                                        "in water",
                                                        "land",
                                                        "land",
                                                        "land"],
        MethodName= ["CML-IA baseline"],
#       Normalization = ["World 2000"],
#       Characterization_Factor =["Abiotic depletion","Global warming (GWP100a)","Human toxicity",
        "Terrestrial ecotoxicity"],
        Characterization_Factor =[
        "Photochemical oxidation",
        #"Photochemical",
        "Human toxicity",
        "Abiotic depletion",
        "Eutrophication",
        "Abiotic depletion (fossil fuels)",
        "Marine aquatic ecotoxicity",
        "Ozone layer depletion (ODP)",
        "Terrestrial ecotoxicity",
        "Acidification",
        "Fresh water aquatic ecotox.",
        "Global warming (GWP100a)"
        ],
        MethodologyType=["attributional"], #consequential
   #AllocationType=["mass"],
        AllocationType=["mass"] #ou “energy” ou “economic”
        );
```

```
NoComps as Integer(Default = 5);

DEVICES
lca1        as LCABasic;

VARIABLES

# LCA from EMSO_OLCA
error(lca1.ncf) as Real;
r_exp(lca1.ncf) as Real;
r(lca1.ncf) as Real;
Mape as Real;

# LCA calculated from Ef and CF functions
LCA_calc(lca1.ncf) as Real;
error2(lca1.ncf) as Real;
Mape2 as Real;

#Percentual of each factor of impact (Inputs and Elementary Flows)
LCA(lca1.ncf,lca1.ni+lca1.ne) as Real;
LCA_p(lca1.ncf,lca1.ni+lca1.ne) as Real;
error_p(lca1.ncf,lca1.ni+lca1.ne) as Real;
MSE_p(lca1.ncf) as Real;

SET
lca1.ni=31;
lca1.ncf=11;
lca1.no=1;
lca1.ne=14;
lca1.nc=0;
lca1.alloc_par(1)=1;

SPECIFY
r_exp=[0.00169359265864469,0.250187125240996, 4.29159397099369E-06, 0.00740822013633725,
3.28619288783233, 257.254945653105, 1.96253443755379E-08, 0.00109230271205232, 0.0139456923021402,
0.138824023835402, 0.0651460519858197];

EQUATIONS

#Test of function LCA with openLCA data
r=[lca1.r(1,:),lca1.r(2,:), lca1.r(3,:), lca1.r(4,:), lca1.r(5,:), lca1.r(6,:), lca1.r(7,:), lca1.r(8,:), lca1.r(9,:),
lca1.r(10,:), lca1.r(11,:)];
error=abs(r_exp-r)/r_exp*100;
Mape=sum(error)/lca1.ncf;

#test of the Ef and CF functions
LCA_calc=sumt(lca1.ef*lca1.emission_values)+sumt(lca1.cf*lca1.input_values);
error2=abs(r-LCA_calc)/r*100;
Mape2=sum(error2)/lca1.ncf;


#test of the percentual
LCA(:,1:lca1.ni)=lca1.cf*lca1.input_values;
LCA(:,lca1.ni+1:lca1.ni+lca1.ne)=lca1.ef*lca1.emission_values;

for i in [1:lca1.ncf] do
        LCA_p(i,:)=LCA(i,:)/LCA_calc(i);
        error_p(i,:)=(lca1.p(i,:)-LCA_p(i,:)); #there is zero in the values so no division is made
        MSE_p(i)=sum(error_p(i,:)^2)/(lca1.ni+lca1.ne);
end
```

```
SPECIFY


#lca1.input_values(3)= (sin(0.1*time*'1/s'*'rad')*12.5923);
lca1.input_values(1)              =          2.46E-07;
lca1.input_values(2) =          4.22E-06;
lca1.input_values(3) =          12.5923;
lca1.input_values(4) =          4.18E-06;
lca1.input_values(5) =          0.0001637;
lca1.input_values(6) =          0.00290882;
lca1.input_values(7) =          5.97E-06;
lca1.input_values(8) =          8.20E-05;
lca1.input_values(9) =          0.0080213;
lca1.input_values(10) =         4.58E-07;
lca1.input_values(11) =         6.76E-05;
lca1.input_values(12) =         3.60E-05;
lca1.input_values(13) =         1.18E-05;
lca1.input_values(14) =         9.56E-05;
lca1.input_values(15) =         7.39E-06;
lca1.input_values(16) =         0.00115975;
lca1.input_values(17) =         0.001637;
lca1.input_values(18) =         6.57E-06;
lca1.input_values(19) =         9.57E-05;
lca1.input_values(20) =         2.19E-06;
lca1.input_values(21) =         2.59E-06;
lca1.input_values(22) =         4.76E-06;
lca1.input_values(23) =         2.46E-07;
lca1.input_values(24) =         7.63E-07;
lca1.input_values(25) =         0.000853758;
lca1.input_values(26) =         0.000114842;
lca1.input_values(27) =         2.56E-07;
lca1.input_values(28) =         1.40E-06;
lca1.input_values(29) =         2.33E-06;
lca1.input_values(30) =         0.00661096;
lca1.input_values(31) =         8.45E-06;

lca1.emission_values(1)=        0.000798352;
lca1.emission_values(2)=        0.00159922;
lca1.emission_values(3)=        9.85E-05;
lca1.emission_values(4)=        3.09557;
lca1.emission_values(5)=        0.000583024;
lca1.emission_values(6)=        0.001637;
lca1.emission_values(7)=        7.78E-05;
lca1.emission_values(8)=        0.00141034;
lca1.emission_values(9)=        7.56E-05;
lca1.emission_values(10)=0.00141034;
lca1.emission_values(11)=       0.0188885;
lca1.emission_values(12)=       8.06E-07;
lca1.emission_values(13)=       4.11E-05;
lca1.emission_values(14)=       8.06E-07;

lca1.output_values(1)= 1;


OPTIONS
Dynamic = false;
TimeStep = 1;
TimeEnd = 100;
TimeUnit = 's';
Integration = "original";
NLASolver(
        File = "nlasolver", #sundials, nlasolver
        RelativeAccuracy = 1e-6,
```

```
                AbsoluteAccuracy = 1e-7,
                MaxIterations = 100
        );
        DAESolver(
                File = "sundials",
                RelativeAccuracy = 1e-6,
                AbsoluteAccuracy = 1e-7,
                EventAccuracy = 1e-2
        );
end


Model LCABasic
        PARAMETERS
outer NoComps as Integer;
outer obj as Plugin(Type="EMSO_OLCA");

        ni as Integer (Brief="Number of Inputs", Default=2);
  ncf as Integer (Brief="Number of Characterization Factors", Default=3);
        no as Integer (Brief= "Number of Outputs", Default=2);
        ne as Integer (Brief= "Number of ElementaryFlows (emissions+resources)", Default=2);
        nc as Integer (Brief= "Number of Consequential Input", Default=0);
        alloc_par(no) as Real (Default=1, Lower=1e-5);


        VARIABLES

        cf(ncf, ni) as Real;
        ef(ncf, ne) as Real;
        r (ncf, no) as Real;
        p (ncf, ni+ne+nc) as Real;

        input_values(ni) as Real (Default=1);
        emission_values(ne) as Real(Default=1);
        output_values(no) as Real (Default=1, Lower=1e-5);

        EQUATIONS

        [ef(1,:),ef(2,:),ef(3,:),ef(4,:),ef(5,:),ef(6,:),ef(7,:),ef(8,:),ef(9,:),ef(10,:),ef(11,:)] = obj.EmissionFactor();

        [cf(1,:),cf(2,:),cf(3,:),cf(4,:),cf(5,:),cf(6,:),cf(7,:),cf(8,:),cf(9,:),cf(10,:),cf(11,:)]  = obj.CharacterizationFactor();

        # keep input values and emissions values in kg
        [r(1,:), r(2,:), r(3,:), r(4,:), r(5,:), r(6,:), r(7,:), r(8,:), r(9,:), r(10,:), r(11,:)] =
                                                                                obj.LcaCalc(input_values,
emission_values, output_values, alloc_par);

        [p(1,:), p(2,:), p(3,:), p(4,:), p(5,:), p(6,:), p(7,:), p(8,:), p(9,:), p(10,:), p(11,:)] =

        obj.LcaPercentual(input_values, emission_values, output_values, alloc_par);

end
```