NI-KOP Author: Mykyta Boiko CTU login: boikomyk Report #5 Popis problému Dána Booleova formule F o n proměnných, X=(x1, x2, ..., xn) v konjunktivní normální formě (CNF). Dále jsou dány celočíselné kladné váhy těchto n proměnných  $\mathbf{W}=(w1, w2, ..., wn)$ . Nalezněte přiřazení  $\mathbf{Y}=(y1, y2, ..., yn)$  proměnných x1, x2, ..., xn, takové, že  $\mathbf{F}(Y)=1$  a součet S vah proměnných nastavených do 1 je maximální. Výběr algoritmu Byly nabídnuté následující heuristické algoritmy: Simulované ochlazování Genetické algoritmy Tabu search Pro řešení problému vážené splnitelnosti booleovské formule jsem zvolil heruistiku Simulované ochlazování. Simulované ochlazování Pseudokód (přednaska č.8, str.14) Počáteční teplota  $\rightarrow$  T; Ø→best; while (!frozen (T, ...)) { while (!equilibrium (...)) { state = try(state); if (state.better(best)) state → best; = cool(T, ...);Simulované ochlazování je iterativní heruistika. Snaží se určitého počátečního řešení postupně dopracovat k řešení co nejlepšímu. Zajimova vlasnost algoritmu je, že se snaží v jednotlivých iteracích uniknout z lokálního minima tím, že může dočasně přijmout horší výsledek. Algoritmus nezaručuje nalezení ideálně optimálního výsledků, přesto může za konstantní počet kroku najít docela příjemný výsledek. Základní parametry algoritmu jsou: počáteční stav počáteční teplota • koeficient ochlazování  $\in [0.8, 0.99]$ • koncová(/minimální) teplota • equilibrium Na začátku algoritmus dostane počáteční stav, t.j. libovolnou konfiguraci, která je řešením. Algoritmus běží, dokud z počáteční teploty nedosáhne minimální teploty, ke které se dostává tak, že postupně násobí aktuální teplotu koeficientem ochlazování (aby aktuální teplota se zmenšovala). V algoritmu zmíněnou funkcionalitu implementuje fukce  $\mathbf{frozen}()$  (vnější cyklus). Ochlazování teploty je implementované funkci **cool()** na konci vnějšího cylku. Vnitřní cyklus je implementován funkci **equilibrium()**, která definuje pevný počet kroku, t.j. počet iteraci na daně teplotě, rovný:  $N_{CelkovyPocetLiteral\mathring{\mathbf{u}}}*equilibrium.$ V každém kroku vnitřního cyklu je volaná funkce  $\mathbf{try}()$ . Unvitr této funci generuje se sousední stav, t.j. sousední řešení(/konfigurace), která se liší jenom inverzi hanodneho bitů aktuálního stavu(t.j změna ohodnocení nějakého literalu, nastavení jej na True nebo False). Pokud je sousední stav lepší než aktuální, tak je vždy přijat. Pokud není, tak bude přijat na základě pravděpodobností danou formuli:  $\operatorname{Pravd\check{e}podobnost}\ \operatorname{p\check{r}ijeti} = e^{-rac{CenaSousednihoStavu-AktualniCena}{AktualniTeplota}}$ Tato pravděpodobnost bude porovnána s náhodným číslem z intervalu [0,1]. Z toho vlastně plynou následující zákonitosti: čím je horší řešení, tím je menší pravděpodobnost, že bude přijato. Avšak, čím je větší aktuální teplota, tím je větší šanci toho, že poměrně horší řešení bude přijato. Tohle právě znázorňuje princip uniknutí z lokálního minima. Po ukončení vnitřního cyklu je snížena aktuální teplota. Pokud se došlo k minimální teplotě, tak algoritmus se zastaví a vrátí nalezeny výsledek. Algoritmus je do detailů popsaný a okomentovaný přímo v kódu. Zkrácený kód: # repeat this process until the current temperature is less than the final temperature while not self.\_frozen(): iteration = 0# stay on this temperature for a while (equilibrium) while self.\_equilibrium(iteration): # get a random neighbor of the current state self.state = self.\_try() if self.state.better(best\_state) and self.state.is\_valid(): # update best state best\_state = deepcopy(self.state) # increment counters iteration += 1 # reduce the temperature on each iteration self.current\_temperature = self.\_cool() Modifikace oproti problému bathou Důležitá modifikace oproti problémů batohu je způsob výpočtu ohodnocení stavu. Základ je následující: součet vah literalů, které jsou ohodnoceny 1. Primární cíl maximalizace tohoto součtu. Avask spočívá v tom velký problém, že algoritmus by vůbec nehledal správné řešení, ale skončil by se všemi literaly nastavenými na 1. Prozkoumal jsem spoustu různých zdrojů a experimentálně zkoušel jsem různé přístupy jak správné definovat cenovou funkci pro problém vážené splnitelnosti booleovské formule. Kazdy z přístupů rychle probereme. cena ovlivněná počtem splněných klauzulí def cost\_affected\_by\_count\_of\_satisfied\_clauses(self) -> int: Returns current state's cost: total weight is multiplied by satisfied clauses count weight = self.sat\_problem.weight(self.configuration) return int(weight \* self.sat\_problem.true\_clauses\_cnt(self.configuration)) Cena stavu =  $Vaha \times PočetSplněnýchKlauzul$ í Výsledná cena v tomto případě byla definována součinem váhy a počtem splněných splněných klauzuli. Tímto přístupem jsem chtěl zvýhodňovat řešení, co mají větší počet splněných klauzuli. Avšak tohle řešení nevždy udávala příjemné výsledky. Proto jsem pokračoval zkoušet další přístupy. A chtěl jsem zkusit úplně obrácený přístup. - cena ovlivněná koefficientem pro nesplnene klauzuli def cost affected by unsatisfied penalty coefficient(self, unsatisfied penalty coefficient: int) -> int: Returns current state's cost: the number of unsatisfied clauses multiplied by maximum weight over all literals and multiplied by unsatisfied\_penalty\_coefficient and then subtracted from the total weight weight = self.sat problem.weight(self.configuration) unsatisfied\_clauses\_cnt = self.sat\_problem.clauses\_cnt self.sat\_problem.true\_clauses\_cnt(self.configuration) return int(weight - (unsatisfied\_clauses\_cnt \* max(self.sat\_problem.weights) \* unsatisfied\_penalty\_coefficient))  ${\rm Cena\ stavu} = Vaha - (Po\check{c}etNespln\check{c}n\acute{c}chKlauzul\acute{u}\times Maximaln\acute{u}VahaLiteralu\times KoeficientemNeuspokojen\acute{u})$ Výsledná cena byla definována rozdílem mezi vahou a počtem nesplněných klauzulí vynásobeným maximální váhou přes všechny literály a vynásobenym "koeficientem neuspokojení". Hodnotu koefficientu jsem volil experementalne z intervalu [1, 10]. Dá se říct, že tahle implementace už vedla na relativně dobře výsledky. Avšak chtěl jsem zkusit ještě další přístupy, resp. přístup který by bral do úvahy jestli je formule splněna nebo ne. - cena ovlivněná penalty koefficientem def cost\_affected\_by\_penalty\_coefficient(self, penalty\_coefficient: int) -> int: Returns current state's cost affected by penalty coefficient weight = self.sat\_problem.weight(self.configuration) # F(configuration) = 1if self.sat\_problem.is\_solution(self.configuration): return weight \* penalty\_coefficient # F(configuration) != 1 return int(weight \* pow(self.sat\_problem.true\_clauses\_ratio(self.configuration), penalty\_coefficient)) Cena stavu(F=1) =  $Vaha \times PenaltyKoefficient$  $\texttt{Cena stavu}(\texttt{F}! \texttt{=} 1) = Vaha \times Pom \check{\texttt{e}}rSpln \check{\texttt{e}}n \acute{\texttt{y}}chKlauzul \acute{\texttt{i}}^{PenaltyKoefficient}$ Stejně jak u předchozího návrhu přidal jsem závislost ceny na novém koefficientu. Avask teď se rozlišovaly případy pro splněnou formule a ne. V případě F=1, tak výsledná cena byla váha vynásobená penalty koefficientem. V opracnem případě váha vynásobená poměrem splněných klauzuli na mocninu penalty koefficientu. Tahle implementace dle mě udávala nejpříjemnější výsledky. Hodnotu koefficientu jsem volil experementalne (bude uvedeno dále). Všechny výš popsané implementace byly použité ve funcki  $\mathbf{better}()$ . Po několika experimentech jsem se rozhodl použít v finálním programu poslední variantu, protože ve většině případů udávala nejlepší výsledky. Předzpracování a generování dát Dále budou definované pomůcné funcke pro generování/načtení testovacích(vstupních) dát, řešení a vyklresleni výsledků experimentů. In [264... # import scope import numpy as np import pandas as pd from io import StringIO import matplotlib import seaborn as sns import matplotlib.pyplot as plt from typing import Dict from prettytable import PrettyTable %matplotlib inline In [14]: # path to solver executor SOLVER = '.../.../sat/main.py' !python3.9 \$SOLVER --help | tail -n 20 Process input problem instances. optional arguments: -h, --help show this help message and exit -in INPUT, --input INPUT paste path to input file -ref REFERENCE, --reference REFERENCE paste path to file containing solution/reference -state generator {InitialStateGenerator.all false,InitialStateGenerator.random}, --state generator {InitialSt ateGenerator.all false,InitialStateGenerator.random} -initial\_temperature INITIAL\_TEMPERATURE, --initial\_temperature INITIAL\_TEMPERATURE initial temperature -cooling coefficient COOLING COEFFICIENT, --cooling coefficient COOLING COEFFICIENT cooling coefficient -minimal temperature MINIMAL TEMPERATURE, --minimal temperature MINIMAL TEMPERATURE minimal temperature -equilibrium coefficient EQUILIBRIUM COEFFICIENT, --equilibrium coefficient EQUILIBRIUM COEFFICIENT equilibrium coefficient -penalty coefficient PENALTY COEFFICIENT, --penalty coefficient PENALTY COEFFICIENT equilibrium coefficient -b, --benchmark display benchmark Základní nastavení parametrů algoritmu: 1 Počáteční teplota 100 2 Koeficient ochlazování 3 Minimální teplota 10 4 Equilibrium 5 Generátor počátečního stavu all\_false Přidal jsem 5-tý parameter do algoritmu, který vlastně říká jak bude generovaný počáteční stav. Pro tyto účely jsem naimplementoval 2 možnosti: • all\_false: ohodnocení všech literalu nastaveno na False random: náhodně ohodnocení literalů Co se tyce -penalty\_coefficient, vysvětlení, jak zjistit správnou hodnotu tohoto parametru, bude uvedeno dále. In [156.. # common constants scope INITIAL TEMPERATURE = 100 COOLING COEFFICIENT = 0.9 MINIMAL TEMPERATURE = 10 EQUILIBRIUM = 3 INITIAL STATE GENERATOR = 'all false' DEFAULT ARGS = { '-initial temperature' : INITIAL TEMPERATURE, '-cooling coefficient' : COOLING COEFFICIENT, '-minimal temperature' : MINIMAL TEMPERATURE, '-equilibrium coefficient': EQUILIBRIUM, : INITIAL STATE GENERATOR, '-state generator' '-penalty\_coefficient' : 5 # An explanation of finding the correct value for this parameter will be expl V ramci zadani byly poskytnuty sady zkušebních instanci wuf-M vcetne optimalnich reseni pro nektere instanci. !ls -la '../../sat/data/wuf-M/' | tail -n 5 drwxr-xr-x@ 1002 user staff - 32064 Dec 7 2019 wuf20-78-M/ -rw-r--r-@ 1 user staff - 73619 Dec 7 2019 wuf20-78-M-opt.dat drwxr-xr-x@ 1002 user staff - 32064 Dec 7 2019 wuf50-201-M/
-rw-r--r-@ 1 user staff - 5948 Dec 7 2019 wuf50-201-M-opt.dat
drwxr-xr-x@ 102 user staff - 3264 Dec 7 2019 wuf75-310-M/ Potřebujeme měřit relativní chybu, proto měření budeme provádět pouze na instancích pro které existuje příslušné referenční řešení. Soustředíme hlavně na instancích ze souboru wuf50-201-M, kde každá instance má: 50 literalů 201 klauzulí Všechny měření budeme provádět na 10 instancích z tohoto souboru. Výsledky budou zprůmerovany. Takže zvolíme 10 náhodných souboru dle referenčního souboru. Bohužel jsem omezený výpočetním výkonem svého počítače a nemůžu čekat více než několik hodin na zpracování vstupních instanci, proto omezil jsem se pouze na 10 instancích. first 10 files = !head -10 \$PATH TO REFERENCE | cut -d' ' -f1 PATH TO REFERENCE = f'../../sat/data/wuf-M/wuf50-201-M-opt.dat' INPUT FILES = [ f'w{input file name}.mwcnf' for input file name in first 10 files.n.split('\n') print(\*INPUT FILES) PATH\_TO\_INPUT\_FILES = f'.../.../sat/data/wuf-M/wuf50-201-M/' wuf50-0123.mwcnf wuf50-012.mwcnf wuf50-0133.mwcnf wuf50-0139.mwcnf wuf50-0241.mwcnf wuf50-024.mwcnf wuf50-0274. mwcnf wuf50-0369.mwcnf wuf50-036.mwcnf wuf50-0405.mwcnf # prepare help functions for data predprocessing def read and store buffer input to df(buffer): # read csv input df = pd.read table(buffer, delimiter="\t", index col=False, low memory=False) return df def solve instance using simulated annealing( parameter\_with\_range\_to\_test: Dict[str,str], args: Dict[str,str] = DEFAULT ARGS, log events = True ): # remove from args tesing param parameter name = next(iter(parameter with range to test.keys())) del args[parameter name] # get valid args str args str = " ".join(map(lambda arg: f"{arg[0]}={arg[1]}", args.items())) # prepare storage for algorithms outputs algorithm outputs = [] for parameter value in next(iter(parameter with range to test.values())): if log events: print(f'\x1b[1;32m[INFO]\x1b[0m: testing parameter \x1b[1;31m {parameter name}={paramete # prepare args (inlcuding param to test) args with test param = f'{args str} {parameter name}={parameter value}' if log events: print(f' - Solving input instances.. :', end='') for indx, input file name in enumerate(INPUT FILES): # ala progress bar if log events: print(f"{input file name}, ", end="") if indx != len(INPUT FILES)-1 else print(f"{ir input file = f'{PATH TO INPUT FILES}{input file name}' # solve input instace and obtain solution solution = !python3.9 \$SOLVER -in=\$input file -ref=\$PATH TO REFERENCE -b \$args with test param # store solution output to dataframe df = read and store buffer input to df( buffer=StringIO(solution.n) # update dataframe with corresponding column representing testing parameter df[parameter name]=parameter value algorithm outputs.append(df) # concatenate all dataframes to single dataframe return pd.concat(algorithm outputs).reset index() def plot basic analytics( algorithm outputs: pd.DataFrame, parameter\_with\_range\_to\_test: Dict[str,str], index column: str, parameter label: str ): # define several pivot tables for different analytic cases spawn pivot table = lambda value, aggfunc: pd.pivot table( data=algorithm outputs, values=value, index=index column, aggfunc=np.mean # Mean for: time, steps, error time mean table = spawn pivot table(value='TIME[ms]', aggfunc=np.mean) steps mean table = spawn pivot table(value='STEPS', aggfunc=np.mean) error mean table = spawn pivot table(value='ERROR', aggfunc=np.mean) df sat analytics = algorithm outputs.copy() data = { 'NOT SAT': [], 'SAT': [], 'OPTIMAL\_SAT': [] parameter name = next(iter(parameter with range to test.keys())) for value in next(iter(parameter\_with\_range\_to\_test.values())): data['NOT SAT'].append(df sat analytics[(df sat analytics['SATISFIED'] == False) & (df sat analytics[pa data['SAT'].append(df sat analytics[(df sat analytics['SATISFIED'] == True) & (df sat analytics[paramet data['OPTIMAL SAT'].append(df sat analytics[ (df sat analytics['SATISFIED'] == True) & (df sat analytics['ERROR'] == 0.0) & (df\_sat\_analytics[parameter\_name] == value) ].count()['SATISFIED']) df sat = pd.DataFrame(data, index = next(iter(parameter with range to test.values()))) \_, axes = plt.subplots(nrows=1, ncols=2, figsize=(22,7)) ax = steps mean table.plot(title=f"Průměr počet kroků VS {parameter label}", ax=axes[0]) ax.set ylabel('Průměr počtu kroků') ax.set\_xlabel(parameter\_label) ax = error mean table.plot(title=f"Průměr relativní chyby VS {parameter label}", ax=axes[1]) ax.set ylabel('Průměr relativní chyby') ax.set\_xlabel(parameter\_label) \_, axes = plt.subplots(nrows=1, ncols=2, figsize=(22,7)) ax = time mean table.plot(title=f"Průměr času v mikrosekundech VS {parameter label}", ax=axes[0]) ax.set ylabel('Průměr času v mikrosekundech') ax.set xlabel(parameter label) ax = df sat.plot(y=["NOT SAT", "SAT", "OPTIMAL SAT"], use index=True, title=f"Počet problemů VS {parameter ax.set ylabel('Počet problémů') ax.set\_xlabel(parameter\_label) plt.show() def display sat table analysis ( parameter with range to test: Dict[str,str], df: pd.DataFrame ): parameter name = next(iter(parameter with range to test.keys())) t = PrettyTable([parameter name, 'Počet není SAT rešení', 'Počet SAT rešení', 'Počet optimalních SAT rešení df analytics = df.copy() for value in next(iter(parameter with range to test.values())): NOT SAT = df analytics[(df analytics['SATISFIED'] == False) & (df analytics[parameter name] == value)] SAT = df analytics[(df analytics['SATISFIED'] == True) & (df analytics[parameter name] == value)].count OPTIMAL SAT = df analytics[ (df analytics['SATISFIED'] == True) & (df analytics['ERROR'] == 0.0) & (df analytics[parameter name] == value) ].count()['SATISFIED'] t.add row([value, NOT SAT, SAT, OPTIMAL SAT]) print(f"\nCelkovy počet problémů {len(INPUT FILES)}") print(t) In [254.. def plot weight evalution history( algorithm output: pd.DataFrame, parameter column name: str, index values ): index values = index values[:4] \_, axes = plt.subplots(nrows=2, ncols=2, figsize=(20,11)) x = 0y = 0 for index value in index values: dataframe = algorithm output.loc[algorithm output[parameter column name] == index value] price history table = pd.DataFrame(data={ 'WEIGHT HISTORY': dataframe['WEIGHT HISTORY'].iloc[0].split(','), # append STEPS column price history table["STEPS"] = price history table.index + 1 # define several pivot tables for different analytic cases spawn pivot table = lambda value, aggfunc: pd.pivot table( data=price history table, values=value, index='STEPS' aggfunc=np.mean price mean table = spawn pivot table(value='WEIGHT HISTORY', aggfunc=np.mean) ax = price mean table.plot(title=f"{parameter column name}={index value}", ax=axes[x, y]) ax.set ylabel('Vyvoj vahy') ax.set xlabel('Počet kroků/Iterace') **if** y == 1: x = 1y = 0 else: y **+=** 1 plt.suptitle('Vyvoj vahy VS Počet navštívených stavů/kroků',fontsize=18) plt.show() def plot history columns evalution ( algorithm output: pd.DataFrame ): history table = pd.DataFrame(data={ 'WEIGHT': algorithm output['WEIGHT HISTORY'].iloc[0].split(','), COST': algorithm\_output['COSTS\_HISTORY'].iloc[0].split(','), 'TEMPERATURE': algorithm output['TEMPERATURE HISTORY'].iloc[0].split(',') }) # append STEPS column history table["STEPS"] = history table.index + 1 history table["WEIGHT"]=history\_table["WEIGHT"].astype(int) history table["COST"]=history table["COST"].astype(float) history table["TEMPERATURE"]=history table["TEMPERATURE"].astype(float) # define several pivot tables for different analytic cases spawn pivot table = lambda value, aggfunc: pd.pivot table( data=history table, values=value, index='STEPS' aggfunc=np.mean temperatue mean table = spawn pivot table(value='TEMPERATURE', aggfunc=np.mean) \_, axes = plt.subplots(nrows=1, ncols=2, figsize=(15,7)) ax = history table.plot(y=['WEIGHT', 'COST'], x='STEPS', title=f"Vyvoj vahy a ceny VS Počet kroků", ax=axes ax.set ylabel('Vyvoj vahy a ceny') ax.set xlabel('Počet kroků/Iterace') ax = temperatue mean table.plot(title=f"Vyvoj teploty VS Počet kroků", ax=axes[1]) ax.set ylabel('Vyvoj teploty') ax.set xlabel('Počet kroků/Iterace') plt.show() Analýza a interpretace dat Penalty koefficient [-penalty\_coefficient] Nejdřív potřebujeme určit optimální hodnotu parametru penálty koefficientu. Pro testování zvolíme interval [1,10] s krokem 1. args = DEFAULT ARGS.copy() penalty coefficient parameter = { # range -penalty\_coefficient <1, 10> '-penalty coefficient': range(1, 10 + 1, 1) algorithm\_outputs\_penalty\_coefficient = solve\_instance\_using\_simulated\_annealing( parameter with range to test=penalty coefficient parameter, args=args [INFO]: testing parameter -penalty\_coefficient=1 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -penalty\_coefficient=2 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -penalty\_coefficient=3 - Solving input instances. :wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -penalty\_coefficient=4 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -penalty\_coefficient=5 - Solving input instances.. :wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -penalty\_coefficient=6 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -penalty\_coefficient=7 - Solving input instances.. :wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -penalty\_coefficient=8 - Solving input instances.. :wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -penalty\_coefficient=9 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -penalty\_coefficient=10 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. # 1. Display Problems count & Mean relative error plots: df analytics = algorithm outputs penalty coefficient.copy() error mean table = pd.pivot table( data=df analytics, values='ERROR', index='-penalty coefficient', aggfunc=np.mean data = { 'NOT SAT': [], 'SAT': [], 'OPTIMAL SAT': [] parameter name = next(iter(penalty coefficient parameter.keys())) for value in next(iter(penalty coefficient parameter.values())): data['NOT SAT'].append(df analytics[(df analytics['SATISFIED'] == False) & (df analytics[parameter name] == data['SAT'].append(df analytics[(df analytics['SATISFIED'] == True) & (df analytics[parameter name] == valu data['OPTIMAL SAT'].append(df analytics[ (df analytics['SATISFIED'] == True) & (df analytics['ERROR'] == 0.0) & (df analytics[parameter name] == value) ].count()['SATISFIED']) df sat = pd.DataFrame(data, index = next(iter(penalty coefficient parameter.values()))) \_, axes = plt.subplots(nrows=1, ncols=2, figsize=(22,9)) # decrease size of legend plt.rc('legend', fontsize=10) ax = df sat.plot(y=["NOT SAT", "SAT", "OPTIMAL SAT"], use index=True, title=f"Počet problemů VS -penalty coeffi ax.set ylabel('Počet problemů') ax = error mean table.plot(title=f"Průměr relativní chyby VS -penalty coefficient", ax=axes[1]) ax.set ylabel('Průměr relativní chyby') plt.show() # 2. Display sat table analytics: display sat table analysis( parameter with range to test=penalty coefficient parameter, df=algorithm outputs penalty coefficient Počet problemů VS -penalty\_coefficient Průměr relativní chyby VS -penalty\_coefficient - ERROR NOT\_SAT 0.5 SAT - OPTIMAL SAT 8 0.4 0.3 0.1 Celkovy počet problem 10 | -penalty coefficient | Počet není SAT rešení | Počet SAT rešení | Počet optimalních SAT rešení | 8 10 0 10 4 0 3 10 10 0 10 Ω 8 10 10 10 Z provedených měření je vidět, že při nízké hodnotě koefficientu algoritmus nevždy je schopny najít takové ohodnocení literalu, při kterém formule bude splněna. Vidíme, že už od 3 algoritmus vždy najde ohodnocení, pro které formule bude splněna. Dá se říct, že nejspíš při docela velké hodnotě koefficientu algoritmus ukazuje tendenci uváznutí v lokálních optimech, neboť relativní chyba začíná postupně narůstat. U 5 a 6 je největší počet nalezených optimálních SAT řešení. Takže zvolím jako hodnotu koefficientu něco uprostřed, což je 5, neboť: • algoritmus vždy dokáže najít ohodnocení literalu, při kterém formule bude splněna relativní chyba je nejmenší Poznámka: první graf Nám říká, že při zpracování 10 vstupních instancí: NOT\_SAT - pro kolik instanci nebylo nalezené takové ohodnocení, aby formule byla splněna, • SAT - pro kolik instanci bylo nalezené takové ohodnocení, aby formule byla splněna, OPTIMAL\_SAT - nalezené ohodnocení je optimální (t.j. relatvni chyba je nulová) Závislosti kvality řešení a výpočetní náročnosti algoritmu na parametrech Jak již bylo zmíněno dříve, všechna měření se provádí na 10 instancích ze souboru wuf50-201-M. Výsledky jsou zprůmerovany. Ve všech provedených měřeních vždy měníme pouze jeden parametr, ostatní zůstavají fixní. Počáteční teplota [-initial\_temperature] Následující test se zabývá citlivosti na počáteční teplotu **-initial\_temperature**. Pro testování zvolíme interval [0, 1000] s krokem 100. args = DEFAULT ARGS.copy() initial temperature parameter = { # range -initial\_temperature <0, 1000> '-initial temperature': range(0, 1000 + 1, 100) algorithm outputs initial temperature = solve instance using simulated annealing( parameter\_with\_range\_to\_test=initial\_temperature\_parameter, args=args [INFO]: testing parameter -initial temperature=0 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -initial\_temperature=100 - Solving input instances.. :wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -initial\_temperature=200 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -initial\_temperature=300 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -initial\_temperature=400 - Solving input instances.. :wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -initial temperature=500 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -initial\_temperature=600 - Solving input instances.. :wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -initial\_temperature=700 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -initial temperature=800 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -initial\_temperature=900 - Solving input instances..: wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. [INFO]: testing parameter -initial\_temperature=1000 - Solving input instances. :wuf50-0123.mwcnf, wuf50-012.mwcnf, wuf50-0133.mwcnf, wuf50-0139.mwcnf, wuf50-024 1.mwcnf, wuf50-024.mwcnf, wuf50-0274.mwcnf, wuf50-0369.mwcnf, wuf50-036.mwcnf, wuf50-0405.mwcnf. plot basic analytics ( algorithm outputs=algorithm outputs initial temperature, parameter with range to test=initial temperature parameter, index column=next(iter(initial temperature parameter)), parameter label='Počáteční teplota' display sat table analysis ( parameter with range to test=initial temperature parameter, df=algorithm outputs initial temperature Průměr počet kroků VS Počáteční teplota Průměr relativní chyby VS Počáteční teplota STEPS - ERROR 1.0 0.8 5000 2 4000 3000 0.4 2000 0.2 1000 0.0 Počáteční teplota Počáteční teplota Průměr času v mikrosekundech VS Počáteční teplota Počet problemů VS Počáteční teplota TIME[ms] Průměr času v mikrosekundech N SAT 200 800 1000 1000 Počáteční teplota Celkovy počet problémů 10 -initial temperature | Počet není SAT rešení | Počet SAT rešení | Počet optimalních SAT rešení | 10 0 100 0 10 200 10 400 500 10 600 10 3 700 0 10 2 800 0 10 900 0 10 1000 Probereme a provedeme analýzu vykreslených grafů. - Počet navštívených stavů: Z grafu je vidět, že při zvyšování počáteční teploty, roste počet prohledovanych stavů. Stejně tak, stojí za zmínku, že koefficient ochlazování se nemění, proto je logicky, že roste počet kroků abychom se dostali na minimální teplotu. - Relativní chyba: Čím větší hodnota počáteční teploty, tím je větší šance u algorimu opustit oblast lokálního minima, neboť zvyšuje pravděpodobnost přijetí horšího řešení. Relativní chyba se snižuje s vyšší počáteční teplotou a dá se říct, že dosahujeme lepších výsledků. Ale stojí za povšimnutí, že na začátku hodnota relativní chyby významné klesá a pak někde kolem 150 se začíná konstantní vývoj. - Časová složitost: Počáteční teplota, stejně jak minimální teplota, má přímý vliv na počet iteraci vnějšího cyklu. Je vidět, že s věští hodnotou počáteční teploty roste i čas. Nejdříve je růst významný, ale po čase se trochu uklidní. - Počet problémů: Z grafu je vidět, že skoro ideální nastavení počáteční teploty je v rozmezí 500-600, neboť počet optimálních nalezených řešení je největší. Stojí za povšimnutí, že algoritmus pro každou hondotu pocateceni teploty větší než 100 dokáže najít takové ohodnocení folrmule, aby ona byla splněna. Vývoj vahy v zavislosti na počtu navštívených stavů plot weight evalution history( algorithm output=algorithm outputs initial temperature, parameter column name='-initial temperature', index values=[200, 500, 800, 1000]

1.mwcnf, [INFO]: t - Solvin	meter_with_range_teargs  esting parameter g input instances wuf50-024.mwcnf, esting parameter g input instances	-cooling_coeffi .:wuf50-0123.m wuf50-0274.mwcnf -cooling_coeffi .:wuf50-0123.m wuf50-0274.mwcnf -cooling_coeffi .:wuf50-0123.m wuf50-0274.mwcnf -cooling_coeffi .:wuf50-0123.m wuf50-0123.m wuf50-0123.m wuf50-0123.m	.cient=0.8 nwcnf, wuf50-0369 .cient=0.81 nwcnf, wuf50-0369 .cient=0.82 nwcnf, wuf50-0369 .cient=0.83 nwcnf, wuf50-0369 .cient=0.83 nwcnf, wuf50-0369 .cient=0.84 nwcnf, wuf50-084	-012.mwcnf, wu -012.mwcnf, wu -012.mwcnf, wu -012.mwcnf, wu -012.mwcnf, wu -012.mwcnf, wu	wuf50-0133.m af50-036.mwcnf wuf50-0133.m af50-036.mwcnf wuf50-0133.m f50-036.mwcnf wuf50-0133.m af50-036.mwcnf	wenf, wuf50-0405.mwc wenf, wuf50-0139 wenf, wuf50-0405.mwc wenf, wuf50-0405.mwc wenf, wuf50-0139 wenf, wuf50-0405.mwc wenf, wuf50-0405.mwc	enf  O.mwenf, wuenf.  O.mwenf, wuenf.  O.mwenf, wuenf.
[INFO]: t - Solvin 1.mwcnf,	wuf50-024.mwcnf, esting parameter g input instances wuf50-024.mwcnf,	-cooling_coeffi .:wuf50-0123.m wuf50-0274.mwcnf	cient=0.85  nwcnf, wuf50-0369  cient=0.86  nwcnf, wuf50-0369  cient=0.87  nwcnf, wuf50-0369  cient=0.88  nwcnf, wuf50-0369  cient=0.89  nwcnf, wuf50-0369  cient=0.9  nwcnf, wuf50-0369  cient=0.9  nwcnf, wuf50-0369  cient=0.9	-012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww	wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf	wcnf, wuf50-0139, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139	enf
- Solvin 1.mwcnf, [INFO]: t - Solvin	esting parameter g input instances wuf50-024.mwcnf, esting parameter g input instances	:wuf50-0123.m wuf50-0274.mwcnf -cooling_coeffi .:wuf50-0123.m wuf50-0274.mwcnf	wenf, wuf50- 2, wuf50-0369 cient=0.92 wenf, wuf50- 6, wuf50-0369 cient=0.93 wenf, wuf50- 6, wuf50-0369 cient=0.95 wenf, wuf50- 6, wuf50-0369 cient=0.96 wenf, wuf50- 6, wuf50-0369 cient=0.96 wenf, wuf50- cient=0.96 cient=0.96	9.mwcnf, ww -012.mwcnf, 9.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww	wuf50-0133.m wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m	wcnf, wuf50-0405.mwc	enf  O.mwcnf, wucnf.  O.mwcnf, wucnf.  O.mwcnf, wucnf.  O.mwcnf, wucnf.  O.mwcnf, wucnf.
1.mwcnf, [INFO]: t - Solvin 1.mwcnf, [INFO]: t - Solvin 1.mwcnf,  plot_bas: algo: parar inde: parar )  display_s parar	wuf50-024.mwcnf, esting parameter g input instances wuf50-024.mwcnf, esting outputs=algorithm_outputs=algorithm_range_ter_label='Koef:  sat_table_analysis meter_with_range_ter_with	wuf50-0274.mwcnf -cooling_coeffi .:wuf50-0123.m wuf50-0274.mwcnf -cooling_coeffi .:wuf50-0123.m wuf50-0274.mwcnf  orithm_outputs_c to_test=cooling_r(cooling_coeffi icient ochlazová s( to_test=cooling_	cient=0.98 wcnf, wuf50-0369 wcnf, wuf50-0369 cient=0.99 wcnf, wuf50-0369 cooling_coeff coefficient_cient_parame ni'	9.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww -012.mwcnf, ww ficient, parameter, eter)),	wuf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m	wenf, wuf50-0405.mwc wenf, wuf50-0139 wuf50-0405.mwc wenf, wuf50-0139	enf  O.mwenf, wu enf  O.mwenf, wu
35000 - STE 30000 - STE 30000 - STE 25000 - STE 25000 - STE 25000 - STE	Průměr počet krol	ků VS Koeficient ochlazování		0.10 - 0.10 - 0.00 - 0.00 - 0.00 -	Průmě	r relativní chyby VS Koeficien	t ochlazování
0.800  1e8  2.0 -  15 -  10 -  10 -  10 -  10 -	Průměr času v mikroseku	0.900 0.925 0.9 cient ochlazování ndech VS Koeficient ochlazová		10 - 8 - 8 - 2 -	Poč	0 0.875 0.900 Koeficient ochlazování et problemů VS Koeficient och	0.925 0.950 nlazování
+	0.825 0.850 0.875  Koeficie  Description of the problem of the pro	0.900 0.925 0.950 ent ochlazování	+		800 0.825 0.850 +  ní   Počet opt	Koeficient ochlazování	1
- Počet na Z grafu je vi polynomiáln - Relativn	0.93 0.94 0.95 0.96 0.97 0.98 0.99 0.99 de provedeme analýzu  a provedeme analýzu  a vštívených stavi  dět, že s rostoucí hodí. Pro hodnoty blízké  aí chyba:  e relativní chyba vžd	<b>ů:</b> dnotou koeficientu k 1 je už nárůst po	ochlazování ro čtu stavu expo	onenciální.	·		
na intervaled vznikají řeše vznikají řeše vznikají řeše vznikají řeše vznikají řeše vznikají řeše vznikají v rostoucím hodnoty blíz v Počet produkty v počet produkty v v voj vahy v v v v v v v v v v v v v v v v v v v	ch [0.84, 0.85] a 0.90 ení, které nejsou SAT e <b>ložitost:</b> koeficientem ochlaz eké k 1 je už růst skor	6. Pokud koukneme (splnitelné). Celker cování roste i časova ro exponecnialni. i intervaly na kterýc kolem <b>0.94</b> . Nebo navštívených stav	e na graf "Poče m vývoj relativi á narocnot. Ste ch vznikají řeše ť má největší p	et problémů" ní chyby má ejně jak u gra ení, které nej	, tak hned je vidě klesající trend. afu pročtu prozko sou SAT. Jinak vh	et, že právě na těch oumaných stavu, da nodné nastavení ko	to intervaled
paraminde: )  10000 -  8000 -  4000 -	rithm_output=algoneter_column_name= x_values=[0.8, 0.8	='-cooling_coeff 88, 0.94, 0.99]	icient',	t navštívený 10000 - 8000 - hup (0000 -	ch stavů/kroků	-cooling_coefficient=0.	88
2000 - 0 - 0 - 10000 - 8000 - 4000 - 2000 -	Poč	750 1000 1250 et kroků/Iterace g_coefficient=0.94	WEIGHT_HISTORY  1500  WEIGHT_HISTORY  5000	2000 - 0 - 10000 - 8000 - 4000 - 2000 -	0 500	1000 1500 Počet kroků/Iterace -cooling_coefficient=0.	
Následující ( [1,99] s kro  args = DI  minimal_f # rai '-min } algorithm param	<b>teplota [-minin</b> est se zabývá citlivo	sti na hodnotu nast  eter = { erature <1, 99> ': range(1, 99 + _temperature_par	tavené miminal  1, 5)  ameter = sol	Lve_instanc	e_using_simula		ní zvolíme int
- Solvin 1.mwcnf, [INFO]: t - Solvin	esting parameter g input instances wuf50-024.mwcnf,	. : wuf50-0123.m wuf50-0274.mwcnf -minimal_temper . : wuf50-0123.m	wcnf, wuf50- ature=6 wcnf, wuf50-0369 ature=11 wcnf, wuf50-0369 ature=16 wcnf, wuf50-0369 ature=21 wcnf, wuf50-0369 ature=21 wcnf, wuf50-0369 ature=21 wcnf, wuf50-0369 ature=26 wcnf, wuf50-0369	9.mwcnf, wu -012.mwcnf, 9.mwcnf, wu -012.mwcnf, 9.mwcnf, wu -012.mwcnf, 9.mwcnf, wu -012.mwcnf, 9.mwcnf, wu	wuf50-0133.m wuf50-0133.m if50-036.mwcnf wuf50-0133.m if50-036.mwcnf wuf50-0133.m if50-036.mwcnf wuf50-0133.m wuf50-0133.m if50-036.mwcnf	wenf, wuf50-0405.mwc	enf
[INFO]: t - Solvin 1.mwcnf, [INFO]: t - Solvin	wurs0-024.mwcnf, esting parameter g input instances wuf50-024.mwcnf, esting parameter	-minimal_temper . : wuf50-0123.m wuf50-0274.mwcnf -minimal_temper . : wuf50-0123.m	cature=31  nwcnf, wuf50-0369  cature=36  nwcnf, wuf50-0369  cature=41  nwcnf, wuf50-0369  cature=46  nwcnf, wuf50-0369  cature=51  nwcnf, wuf50-0369  cature=56  nwcnf, wuf50-0369	-012.mwcnf, wn	wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf	wcnf, wuf50-0139, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139, wuf50-0405.mwcwcnf, wuf50-0139	enf
- Solvin 1.mwcnf, [INFO]: t - Solvin	g input instances wuf50-024.mwcnf, esting parameter g input instances	. : wuf50-0123.m wuf50-0274.mwcnf -minimal_temper . : wuf50-0123.m	wcnf, wuf50- ature=66 wuf50-0369 ature=71 wcnf, wuf50-0369 ature=76 wcnf, wuf50-0369 ature=81 wcnf, wuf50-0369 ature=81 wcnf, wuf50-0369 ature=86 wcnf, wuf50-0369 ature=86 wcnf, wuf50-0369 ature=86 wcnf, wuf50-0369 ature=91 wcnf, wuf50-0369	9.mwcnf, ww -012.mwcnf, 9.mwcnf, ww -012.mwcnf, 9.mwcnf, ww -012.mwcnf, 9.mwcnf, ww -012.mwcnf, 9.mwcnf, ww -012.mwcnf, 9.mwcnf, ww	wuf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf	wenf, wuf50-0405.mwc wenf, wuf50-0139 wenf, wuf50-0139 wenf, wuf50-0405.mwc wenf, wuf50-0405.mwc wenf, wuf50-0405.mwc wenf, wuf50-0405.mwc wenf, wuf50-0405.mwc wenf, wuf50-0405.mwc wenf, wuf50-0139 wenf, wuf50-0139	enf  O.mwcnf, wucnf.  O.mwcnf, wucnf.  O.mwcnf, wucnf.  O.mwcnf, wucnf.  O.mwcnf, wucnf.
[INFO]: t - Solvin 1.mwcnf,  plot_base algo: parar inde: parar )  display_s parar	wuf50-024.mwcnf, esting parameter g input instances wuf50-024.mwcnf, esting parameter g input instances wuf50-024.mwcnf, esting parameter g input instances wuf50-024.mwcnf, esting putputs=algorithm_outputs=algorithm_outputs_restricted by the control of the cont	-minimal_temper:wuf50-0123.m wuf50-0274.mwcnf  orithm_outputs_m to_test=minimal_ r(minimal_temper mální teplota'  s( to_test=minimal_	cature=96 nwcnf, wuf50-0369 inimal_tempe temperature_ ature_parame temperature_ ure_paramete	-012.mwcnf, wu 9.mwcnf, wu erature_par parameter, eter)),  parameter,	wuf50-0133.minf50-036.mwcnf	wcnf, wuf50-0139	enf
6000 - 5000 - 5000 - 1000 - 1000 - 0 - 0		60 imální teplota undech VS Minimální teplota	STEPS  80 100	0.8 - 0.6 - 0.0 -	ERROR DO 20 Po	40 60 Minimální teplota čet problemů VS Minimální te	80 eplota
Průměr času v mikrosekundech	20 40	60 8 Silní teplota	0 100	Počet problémů	20	NOT_SAT SAT OPTIMAL_SAT  40 Minimální teplota	80
+   -minima +	l_temperature   P		'		1		eení  +    +
- Počet na Je jasný, že Zmenšuje se - Relativn Vývoj relativ	81 86 91 96 1 96 a provedeme analýzu  avštívených stavi se zvýšením minimla e stavový prostor, kte  ní chyba: ní chyby je docela ko e říct, že u teplot se k	ů: ani teploty zastavím erý prohledáváme. onstantní. Vypadá,	ie algoritmus d že na zkoumar	ném intervalu	ı minimální teplo	a nemá vliv na kva	litu řešení. A
neproběhne  - Časová s  Stejně jak u  - Počet pr  Dá se říct, ž  jasný, že u t  Z vykreslené řešení.  Vývoj vahy	ještě dostatek iterac ložitost: grafu počtu navštíve	ci pro nalezení lepš ených stavů se zvýš hodnoty minimální čáteční teplotě bud že ideální hodnota	ích řešení a jso sením minimáln teploty na zvo e klesat počet pro minimální t	ou přijímána ní teplty zme oleném interv nalezených	horší řešení. nšuje se doba vý valu nemají příliš řešení. Třeba u 9	počtu. vysoký vliv na výsl 6 všechna nalezen	edky algoritı a řešení nejs
algo: param	ght_evalution_hist rithm_output=algor meter_column_name= x_values=[6, 36, 7	rithm_outputs_mi ='-minimal_tempe 71, 96]	rature',		ch stavů/kroků	-minimal_temperature=	36
2000 - 0 - 0 - 10000 - V 8000 - V 4000 - 2000 -		2000 2500 3000 et kroků/Iterace Il_temperature=71	WEIGHT_HISTORY 3500 4000	2000 - 0 - 0.04 - 0.02 - 448 0.00 - -0.02 - -0.04 -	o 200 400	600 800 Počet kroků/Iterace -minimal_temperature=	1000 1200 96 — w
Následující i  args = Di  equilibr: # rai '-equ' }	poč  m [-equilibrium  rest se zabývá citlivo  EFAULT_ARGS.copy()  lum_parameter = {     inge -minimal_temper     ilibrium_coeffic:     n_outputs_equilibrium     neter with range for	sti na equilibriu -ec ) erature <1, 99> ient': range(0,	quilibrium_cod  50 + 1, 5)  tance_using_	_simulated_		60 80 Počet kroků/Iterace ne interval $[0,50]$ s	s krokem 5.
[INFO]: t - Solvin 1.mwcnf, [INFO]: t - Solvin	esting parameter g input instances wuf50-024.mwcnf, esting parameter	:wuf50-0123.m wuf50-0274.mwcnf -equilibrium_co .:wuf50-0123.m	wcnf, wuf50-0369 efficient=5 wcnf, wuf50-0369 efficient=10 wcnf, wuf50-0369 efficient=15 wcnf, wuf50-0369 efficient=20 wcnf, wuf50-0369 efficient=20 wcnf, wuf50-0369 efficient=26 wcnf, wuf50-0369	9.mwcnf, wu -012.mwcnf, 9.mwcnf, wu 0 -012.mwcnf, wu 5 -012.mwcnf, wu 0 -012.mwcnf, wu 0 -012.mwcnf, wu	wuf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf wuf50-0133.m uf50-036.mwcnf	wenf, wuf50-0405.mwc wenf, wuf50-0139 wenf, wuf50-0405.mwc wenf, wuf50-0405.mwc wenf, wuf50-0139 wenf, wuf50-0405.mwc wenf, wuf50-0405.mwc	enf  O.mwcnf, wu enf  O.mwcnf, wu enf  O.mwcnf, wu enf
- Solvin 1.mwcnf, [INFO]: t - Solvin 1.mwcnf,	g input instances wuf50-024.mwcnf, esting parameter g input instances wuf50-024.mwcnf,	:wuf50-0123.m wuf50-0274.mwcnf -equilibrium_co:wuf50-0123.m wuf50-0274.mwcnf -equilibrium_co:wuf50-0123.m wuf50-0274.mwcnf -equilibrium_co:wuf50-0123.m wuf50-0274.mwcnf -equilibrium_co:wuf50-0123.m wuf50-0274.mwcnf -equilibrium_co:wuf50-0123.m wuf50-0274.mwcnf -equilibrium_co:wuf50-0123.m	wcnf, wuf50-0369 efficient=30 wcnf, wuf50-0369 efficient=35 wcnf, wuf50-0369 efficient=40 wcnf, wuf50-0369 efficient=45 wcnf, wuf50-0369 efficient=50 wcnf, wuf50-0369 efficient=50 wcnf, wuf50-0369 efficient=50 wcnf, wuf50-0369	-012.mwcnf, wu 0 -012.mwcnf, wu 5 -012.mwcnf, wu 0 -012.mwcnf, wu 5 -012.mwcnf, wu 5 -012.mwcnf, wu 5 -012.mwcnf, wu 6 -012.mwcnf, wu 0 -012.mwcnf, wu 0 -012.mwcnf, wu 0 -012.mwcnf, wu 0 -012.mwcnf,	wuf50-0133.m if50-036.mwcnf wuf50-0133.m if50-036.mwcnf wuf50-0133.m if50-036.mwcnf wuf50-0133.m if50-036.mwcnf wuf50-0133.m if50-036.mwcnf	wenf, wuf50-0405.mwc	enf
algo: parar inde: parar )  display_s parar df=a: )	rithm_outputs=algoneter_with_range_tex_column=next(itex_neter_label='Equipmeter_label='Equipmeter_with_range_tex_column_outputs_column_output	to_test=equilibr r(equilibrium_pa librium'  s( to_test=equilibr	<pre>ium_paramete rameter)),</pre>	er, 10-	F	růměr relativní chyby VS Equ	ilibrium
10000 - 10000	Průměr času v mikro	50 Equilibrium osekundech VS Equilibrium	10 50	- 0.0 - Luguez relativní chyby	0 10	20 30 Equilibrium Počet problemů VS Equilibr	40 ium
2.5 - 2.0 -	no zo Economic de la constanta	30 40 quilibrium  -+	+	Počet SAT 1	in the second of	20 Equilibrium 30 Optimalních SAT	40+ rešení
+	10 15 20 25 30 35 40 45 50 a provedeme analýzu avštívených stavi et z prvního pohledu nitřního cyklu.	ů:		10 10 10 10 10 10 10 10	očtu navštívených	6 4 3 5 4 6 4 4	 
- Počet na	<b>ní chyba:</b> odnotou equilibria ro ch stavů. Je viditelně slost chyby na equilik	e, že equilibrium má priu vypadá jako ex é, co i na grafu poč uli přetížení počítač vy 5 algoritmus vždy vrdit, že s větší hod	značný vliv na ponenciální. Av tu navštívenýc će (v důsledku y dokáže najít i notou equilibri	a relativní ch však od hodr ch stavu. Čas zatížení RAM SAT řešení p ia roste i poč	ybu. S rostoucím noty 5 se relativn sová složitost s vo M nebo něco pod ro zkušební insta et nalezených op	koeficientem equi í chyby skoro nemá elikosti equilibria ro obného). anci. Co se týče nal otimálních řešení, n	libria klesá r éní. este lineárně ezení optima
- Počet na Co by šlo říc provedení v - Relativn S rostoucí h generovaný chyba. Závis - Časová s Vlastně tady vychylka na - Počet pr Z grafu je vi řešení, tak z	c <b>ložitost:</b> v pozorujeme to samo grafu jsou nejspis klo	iáme zároveň u dvo nalezenyh optimáln	ou hodnot equi lích řešení. Aši	libria a to jsc potřebovali	u 10 a 40. Bylo b bychom mít buď r počítač. Výpoč	y logicky říct, že s větší počet zkuser et toho intervalu trv	rostoucí hoo bich instand val kolem 4 h
requility -equility -equil	pozorujeme to same grafu jsou nejspis klu grafu jsou nejspis klu grafu jsou nejspis klu grafu nejde 100% to optimálních řešení meži by narůstat počet prozsah pro hodnoty eme udělat závěr, že zavislosti na počtu grafu nejde na počtu nejde na	tory( rithm_outputs_eq -'-equilibrium_cd 40, 50]	ulky šlo říct, že ů uilibrium, oefficient',		ch stavů/kroků		۷
requility -equility -equil	Počitost:  / pozorujeme to same grafu jsou nejspis klustoblémů:  dět, že už od hodnot grafu nejde 100% tvoptimálních řešení měl by narůstat počet prozsah pro hodnoty eme udělat závěr, že z z z zavislosti na počtu ght_evalution_historithm_output=algorithm_output	tory( rithm_outputs_eq -'-equilibrium_c 40, 50]	ulky šlo říct, že ů uilibrium, oefficient',		ch stavů/kroků	-equilibrium_coefficient= 10000 Počet kroků/Iterace -equilibrium_coefficient=	15000
Probereme :  Počet na :  Co by šlo říc provedení v  Relativi :  S rostoucí h generovaný chyba. Závis  Vlastně tady vychylka na :  Počet provedení, tak z nalezených equilibria mežádat větší i Takže může 10 nebo 40  Vývoj vahy :  Plot_weig algo: paraminde:  )  Volba pod  Mario do	pozorujeme to same grafu jsou nejspis kla grafu jsou nejspis kla grafu jsou nejspis kla grafu jsou nejspis kla grafu nejde 100% to optimálních řešení meži by narůstat počet prozah pro hodnoty eme udělat závěr, že za zavislosti na počtu ght_evalution_histrithm_output=algorateter_column_name a_values=[5, 20, 4]  -equilibri	rium_coefficient=40  vykreslené by tab  navštívených stav  tory ( rithm_outputs_eq e'-equilibrium_c 40, 50]  Vyvoj  rium_coefficient=5	ulky šlo říct, že  ů  uilibrium, oefficient',  vahy VS Počet  — WEIGHT_HISTORY 5000  kolik možnosti	t navštívený  10000 - 8000 - 10000 - 2000 - 10000 - 8000 - 2000 - 0 - 10000 -	0 5000	10000 Počet kroků/Iterace -equilibrium_coefficient: 20000 30000 Počet kroků/Iterace	15000 =50 — w 40000
Probereme and the state of the	Pozorujeme to same grafu jsou nejspis klu poblémů:  dět, že už od hodnot grafu nejde 100% tvoptimálních řešení meži by narůstat počet pozsah pro hodnoty eme udělat závěr, že zavislosti na počtu ght_evalution_histrithm_output=algorateter_column_names a_values=[5, 20, 4]  deter_column_names a_values=[5, 20, 4]	ravštívených stav  tory( rithm_outputs_eq ='-equilibrium_cd 40, 50]  Vyvoj  rium_coefficient=5  rium_coefficient=40  rementoval jsem ně oritmu a počet proz istíme, jestli způsol false', 'random'  YAILABLE_STATES enerator_paramet to_test=state_ge  -state_generato:wuf50-0123.m	ulky šlo říct, že  ů  uilibrium, oefficient',  vahy VS Počet  vahy VS Počet  vahy VS Počet  value (  v	t navštívený  t navštívený  10000 - 8000 - 2000 - 10000 - 8000 - 10000	o sooo stavu. Je stavu má nějaký v wuf50-0133.m	10000 Počet kroků/Iterace -equilibrium_coefficient:  20000 30000 Počet kroků/Iterace  jasný, že volba poc /liv výslednou kvali _annealing(  _annealing(	- w 40000  catechniho s tu řešení.
requilities requilities requilities requilities reprobereme reprob	pozorujeme to same grafu jsou nejspis klu grafu jsou nejspis klu grafu jsou nejspis klu grafu nejde 100% two optimálních řešení meži by narůstat počet prozsah pro hodnoty eme udělat závěr, že z z z zavislosti na počtu ght_evalution_histrithm_output=algorater_column_name ac_values=[5, 20, 4]  pare experimentu a zj grafu produce z zavislosti na počtu ght_evalution_histrithm_output=algorater_column_name ac_values=[5, 20, 4]  pare experimentu a zj grafu produce z zavislosti na počtu ght_evalution_histrithm_output=algorater_column_name ac_values=[5, 20, 4]  produce z zavislosti na počtu ght_evalues=[5, 20, 4]  produce z zavislosti na počtu	ruykreslené by tab  navštívených stav  tory( rithm_outputs_eq ='-equilibrium_c 40, 50]  Vyvoj  rium_coefficient=5  rium_coefficient=40  et kroků/terace  ru  ementoval jsem ně oritmu a počet proz istíme, jestli způsol  false', 'random'  = { VAILABLE_STATES  enerator_paramet to_test=state_ge  -state_generato:wuf50-0123.m wuf50-0274.mwcnf -state_generato:wuf50-0123.m wuf50-0274.mwcnf AILABLE_STATES: generator =", state evalution( algorithm_output	weight History  for a solve in  nerator para  weight History  weight History  weight History  and weight History  weight History  weight History  weight History  weight History  and weight History  weight History  weight History  weight History  weight History  and weight History  weight History  weight History  weight History  and weight History	t navštívený  t navštívený  10000  8000  10000  10000  10000  8000  2000  2000  2000  10000  10000  10000  2000  2000  10000  2000  2000  2000  10000  2000	ečního stavu. Je  stavu má nějaký v  stavu má nějaký v  "="*40)  "="*40)  meter.loc[algo  FALSE =======	10000 Počet kroků/Iterace equilibrium_coefficient:  20000 30000 Počet kroků/Iterace  jasný, že volba poc /liv výslednou kvali annealing (  wcnf, wuf50-0139 wcnf, wuf50-0139 wcnf, wuf50-0405.mwc wcnf, wuf50-0405.mwc	- w 40000  atechniho s tu řešení.  .mwcnf, wu cnf  .mwcnf, wu cnf  tate_gener
Probereme: - Počet na Co by šlo říc provedení v - Relativi S rostoucí h generovaný chyba. Závi - Časová s Vlastně tady vychylka na - Počet pi Z grafu je vi řešení, tak z nalezených equilibria m žádat větší Takže může 10 nebo 40  Vývoj vahy  plot_wei algori inde: )  10000  8000  10000  Age 10000  Age 10000  100000  1	pozorujeme to same grafu jsou nejspis klu grafu jsou nejspis klu grafu jsou nejspis klu grafu nejde 100% two optimálních řešení meži by narůstat počet prozsah pro hodnoty eme udělat závěr, že z z z zavislosti na počtu ght_evalution_histrithm_output=algorater_column_name ac_values=[5, 20, 4]  pare experimentu a zj grafu produce z zavislosti na počtu ght_evalution_histrithm_output=algorater_column_name ac_values=[5, 20, 4]  pare experimentu a zj grafu produce z zavislosti na počtu ght_evalution_histrithm_output=algorater_column_name ac_values=[5, 20, 4]  produce z zavislosti na počtu ght_evalues=[5, 20, 4]  produce z zavislosti na počtu	rowstívených stav  tory( rithm_outputs_eq ='-equilibrium_co 40, 50]  Vyvoj  vyvoj  rium_coefficient=5  rium_coefficient=5  rium_coefficient=40  rium_coefficient=5  rium_coefficient=6  rium_coefficient=6  rium_coefficient=6  rium_coefficient=6  rium_coefficient=6  rium_coefficient=6  rium_coefficient=6  rium_coefficient=6  rium_coefficient=6  rium_coefficient=7  rium_coefficient=6  rium_coefficient=6  rium_coefficient=6  rium_coefficient=7  rium_coefficient=6  rium_co	ulky šlo říct, že ů  uilibrium, oefficient', vahy VS Počet  vahy V	t navštívený  t navštívený  10000 - 8000 - 2	ečního stavu. Je  stavu má nějaký v  stavu má nějaký v  "="*40)  "="*40)  meter.loc[algo  FALSE =======	10000 Počet kroků/Iterace -equilibrium_coefficient:  20000 30000 Počet kroků/Iterace  jasný, že volba poc /liv výslednou kvali  annealing(  wcnf, wuf50-0139 , wuf50-0405.mwc  wcnf, wuf50-0405.mwc  orithm_outputs_s	- w 40000  atechniho s tu řešení.  .mwcnf, wu cnf  .mwcnf, wu cnf  tate_gener
Probereme and the probability of the provided	iložitost:  // pozorujeme to sam: grafu jsou nejspis kli  // poblémů:  dět, že už od hodnot grafu nejde 100% tv optimálních řešení m či by narůstat počet i ozsah pro hodnoty e me udělat závěr, že z  // zavislosti na počtu  ght_evalution_hist cithm_output=algo: neter_column_name c_values=[5, 20, // perediction in the column	z vykreslené by tab  navštívených stav  tory(	ulky šlo říct, že ů  uilibrium, oefficient', vahy VS Počet  vahy V	t navštívený  t navštívený  10000 - 8000 - 10000 - 10000 - 8000 - 10000 - 8000 - 10000	ečního stavu. Je  stavu má nějaký v  wuf50-0133.m  f50-036.mwcnf  wuf50-0133.m  f50-036.mwcnf  "="*40)  meter.loc[algo  TALSE ====================================	10000 Počet kroků/Iterace -equilibrium_coefficient:  20000 30000 Počet kroků/Iterace  jasný, že volba poc /liv výslednou kvali  annealing(  wcnf, wuf50-0139 , wuf50-0405.mwc  wcnf, wuf50-0405.mwc  orithm_outputs_s	atechniho s  tu řešení.  atechniho s  tu řešení.  atechniho s  tu řešení.  tate_gener  tate_gener  tkroků  TEM
Probereme - Počet na Co by šlo říc provedení v - Relativi S rostoucí h generovaný chyba. Závis - Časová s Vlastně tady vychylka na - Počet pi Z grafu je vi řešení, tak z nalezených equilibria m žádat větší r Takže může 10 nebo 40 Vývoj vahy  plot_weic algo: parat inde: )  Vidíme, že v  ** ** ** ** ** ** ** ** ** ** ** ** *	pozorujeme to sam grafu jsou nejspis klu potenia jsou nejspis klu potenia jsou nejspis klu potenia jsou nejspis klu potenia je	z vykreslené by tab  navštívených stav  tory(  ithmoutputs eq  '-equilibrium o  tory(  ithmoutput et rowd/iterace  ithmoutput et rowd/iterace  ithmoutput et rowd/iterace  '-state generato  '-state generato state generato sta	weight history  weight history	t navštívený  10000 - 8000 - 10000 - 1	ečního stavu. Je stavu má nějaký v  ving_simulated  wuf50-0133.m  if50-036.mwcnf  wuf50-0133.m  if50-036.mwcnf  "="*40)  meter.loc[algo  PALSE =======  V  V  V  V  V  V  V  V  V  V  V	1000 Počet kroků/terace equilibrium_coefficient:  20000 Počet kroků/terace  jasný, že volba poc vliv výslednou kvali  annealing (  wcnf, wuf50-0138 wuf50-0405.mwc wcnf, wuf50-0405.mwc prithm_outputs_s  prithm_outputs_s  yvoj teploty VS Poče  u ohodnocených Ti	atechniho satu řešení.  catechniho satu řešení.
Probereme :  - Počet na Co by šlo říc provedení v  - Relativi S rostoucí h generovaný chyba. Závis  - Časová s Vlastně tady vychylka na - Počet provedené vychylka na - Počet provedené vychylka na - Počet provedené vychylka na chyba provedené vychylka na chyba provedené vyvoj vahy vychylka na chyba provedené vyvoj vahy vychylka na chyba provedené vyvoj vahy vyvoj vyvoj vahy vyvoj vyvoj vahy vyv	iložitost:  / pozorujeme to sam grafu jsou nejspis kli  / potenta jsou nejspis  / potenta	z vykreslené by tab  navštívených stav  tory ( rithm_outputs_eq ='-equilibrium_outputs_eq ='-equilibrium_outputs_eq ='-equilibrium_outputs_eq ='-equilibrium_outputs_eq ='-equilibrium_outputs_eq ='-equilibrium_outputs_eq ='-equilibrium_outputs_eq et kroků/terace  rum_coefficient=5   'U  dementoval jsem ně oritmu a počet prozistíme, jestli způsol false', 'random' et kroků/terace  =-state_generatc:wuf50-0123.n wuf50-0123.n wuf50-0124.m wuff0-0123.n wuf50-0124.m wuff0-0123.n wuf50-0124.m wuff0-0123.n wuf50-0124.m wuff0-0123.n wuf50-0124.m wuff0-0123.n	weight history  weight history	to navštívený  to navštívený  to navštívený  to navštívený  10000 -  8000 -  10000 -  8000 -  10000 -  8000 -  10000 -	ečního stavu. Je stavu má nějaký v  sing_simulated  wuf50-0133.m f50-036.mwcnf  wuf50-0133.m f50-036.mwcnf  "="*40) meter.loc[algo PALSE ========  V  O 500  NA =========  V  O sono  Na ==========  V  O sono  Na ===================================	20000 30000 Počet kroků/terace equilibrium_coefficient:  annealing (  wonf, wuf50-0138, wuf50-0405.mwc wonf, wuf50-0405.mwc orithm_outputs_s  prithm_outputs_s  yvoj teploty VS Poče  u ohodnocených Toho v sekci "Modii ttečního stavu. Je thodnocení literalů".  e z již minulých experter (  eter [	atechniho statechniho statechniho state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the state generation in the state generation is a state generation in the stat
Probereme:  - Počet na Co by šlo říc provedení v  - Relativi S rostoucí h genyou záví - Časová s  Vlastně tady vychylka na - Počet pi Z grafu je vi řešení, tak z nalezených equilibria m žádat větší Takže může 10 nebo 40  Vývoj vahy  plotuse algorith args = Di state plot state plot  Source  - Na Source Source Source  - Vidíme, že v  - Solvin 1. mycnf, I - Solvin I - Solv	pozorujeme to samgrafu jsou nejspis klusoblémů:  doblémů:  doty že už od hodnot grafu nejde 100% tv optimálních řešení m šu by narůstat počet nozsah pro hodnoty e me udělat závěř, že z v zavislosti na počtu  ght_evalution_nistithm_output=algo: meter_column_name: c_values=[5, 20, 1]  setter_doumentur a zj  par experimentu zj  par experimentu a zj  par	z vykreslené by tab  navštívených stav  tory( rithm_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq et kroků/terace "-equilibrium_outputs_eq et kroků/terace "-equilibrium_outputs_eq et kroků/terace "-equilibrium_outputs_eq et kroků/terace "-state_generator_state_generator_state_generator=", state_generator=", state_generato	weight history  at a generate  satate g	t navštívený  t navštívený  t navštívený  10000 -  8000 -  10000 -  10000 -  8000 -  10000 -	de stavu má nějaký v  stavu má n	Jodoo Pocet kroků/Iterace equilibrium_coefficient:  Joseph Počet kroků/Iterace equilibrium_coefficient:  Jannealing ( Joseph Výslednou kvali  Jannealing ( Joseph Výslednou kvali  Jannealing ( Joseph Výslednou kvali  Joseph	atechniho setu řešení.  catechniho setu řešení.
Probeneme  Počet na Co by šlo říc provedení v  Relativi S rostoucí h genyovaný chyba. Závi  - Časová s  Vlastně tady vychylka na - Počet pi Z grafu je vi řešení, tak z nalezených equilibria m žádat větší Takže může 10 nebo 40  Vývoj vahy  plot weic algo: parar inde: ) 2000  Masono parar ind	pozorujeme to sam grafu jsou nejspis kli pozorujeme to sam grafu jsou nejspis kli poblémů:  dět, že už od hodnot grafu nejde 100% tv optimálních řešení mišl by narůstat počet pozsah pro hodnoty eme udělat závěr, že z do v zavislosti na počtu jste v zavislosti na počtu na p	z vykreslené by tab  navštívených stav  tory( rithm_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq "-equilibrium_outputs_eq et kroků/terace "-equilibrium_outputs_eq et kroků/terace "-equilibrium_outputs_eq et kroků/terace "-equilibrium_outputs_eq et kroků/terace "-state_generator_state_generator_state_generator=", state_generator=", state_generato	weight history  at a generate  satate g	t navštívený  t navštívený  t navštívený  toooo -  8000 -  10000 -	o 500  ečního stavu. Je  stavu má nějaký v  stavu m	Jodoo Pocet kroků/Iterace equilibrium_coefficient:  Joseph Počet kroků/Iterace equilibrium_coefficient:  Jannealing ( Joseph Výslednou kvali  Jannealing ( Joseph Výslednou kvali  Jannealing ( Joseph Výslednou kvali  Joseph	attechniho si tu řešení.  attechniho si atte
Probereme Prober	idožitost:  v pozorujeme to sam grafu jsou nejspis kli voblémů:  dět, že už od hodnot grafu nejde 100% tv optimálních řešení m ši by narůstat počet i ozsah pro hodnoty e me udělat závěr, že závislosti na počtu v zavislosti na požtu na požtu v zavislosti na požtu na	z vykreslené by tab  navštívených stav  tory ( intim_outputs_eq  tory ( intim_outputs_eq  do, 50]  Vyvoj  vyvoj  ium_coefficient=5   et ivosúnteriace  ium_coefficient=5  et ivosúnteriace  ium_coefficient=40   ium_coefficient=40  ium_coefficient=5  ium_coefficient=5  ium_coefficient=5  ium_coefficient=5  ium_coefficient=40  ium_coefficient=40  ium_coefficient=40  ium_coefficient=40  ium_coefficient=40  ium_coefficient=40  ium_coefficient=40  ium_coefficient=40  ium_coefficient=40  ium_coefficient=5  ium_coefficient=6  ium_c	weight wift of a state generated at generate	t navštívený  t navštívený  t navštívený  10000 -  8000 -  10000 -	ečního stavu. Je  stavu má nějaký v  stavu má nějak	poet kroků/terace equilibrium_coefficient:  20000	atechniho si tu řešení.  atechniho si at