FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Evaluace frameworku SEAGE
Student:	David Omrai
Vedoucí:	Ing. Mgr. Ladislava Smítková Janků, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Cílem této bakalářské práce je návrh a implementace propojení evaluátoru hyper-heuristik HyFlex a optimalizačního frameworku SEAGE a zhodnocení optimalizačního frameworku SEAGE z pohledu aktuálního stavu výzkumu v oblasti hyper-heuristik.

1. Seznamte se s problematikou hyper-heuristik, zpracujte rešerši aktuálního stavu výzkumu v této oblasti.

2. Seznamte se s frameworkem SEAGE a s testovacím prostředím HyFlex.

3. Navrhněte a implementujte způsob propojení frameworku SEAGE a testovacího prostředí HyFlex.

4. Navrhněte a realizujte experimenty pro evaluaci hyper-heuristiky/meta-heuristik, prezentujte výsledky experimentů.

5. Navrhněte vylepšení hyper-heuristiky, implementujte toto vylepšení a otestujte ho.

6. Zhodnoďte framework SEAGE v kontextu aktuálního stavu výzkumu v oblasti hyper-heuristik.

Elektronicky schválil/a Ing. Karel Klouda, Ph.D. dne 28. ledna 2021 v Praze.

Bakalářská práce

EVALUACE FRAMEWORKU SEAGE

David Omrai

Fakulta informačních technologií ČVUT v Praze
Katedra teoretické informatiky
Vedoucí: Ing. Mgr. Ladislava Smítková Janků, Ph.D
13. května 2021

Odkaz na tuto práci: David Omrai. *Evaluace frameworku SEAGE*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych především poděkovat Ing. Mgr. Ladislavě Smítků Jánků, Ph.D za odborné vedení, trpělivost a ochotu, kterou mi v průběhu tvorby této bakalářské práce věnovala. Dále bych rád poděkoval Ing. Richardu Málkovi, za jeho nesčetné rady během spolupráce na vývoji frameworku SEAGE, Vojtěchu Tulekovi za pomoc při korekci textu a v neposlední řadě i své rodině, která mi byla oporou po celou dobu tvorby této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2021 .....

Abstrakt

Tato práce je zaměřena na zhodnocení optimalizačního frameworku *SEAGE* z pohledu aktuálního stavu výzkumu v oblasti hyper-heuristik, a to s využitím nové metriky, která algoritmy ohodnocuje objektivně dle kvality jejich řešení instancí optimalizačních problémů.

Porovnání je provedeno na jednom systému mezi již implementovanými heuristikami ve frameworku *SEAGE* a hyper-heuristikami vytvořenými pro mezinárodní výzvu *CHeSC2011*.

Na začátku práce popisujeme naši motivaci, cíle a dále také rešerši aktuálního stavu výzkumu v této oblasti, díky které jsme objevili framework *HyFlex* společně s hyper-heuristikami účastníků výzvy *CHeSC2011*.

Během implementace jsme narazili na řadu problémů, které nás odradili od využití již vytvořeného evaluátoru a metriky ve frameworku *HyFlex*. Jejich nevýhody v práci blíže rozebíráme a představujeme metriku i evaluátor nový, který zmíněné problémy řeší a zároveň přináší určitou formu vylepšení. Pro představení snadnosti tvorby hyper-heuristik v prostředí *SEAGE* jsme vytvořili takovou, která jednoduše využívá vylepšení objektivního ohodnocení řešení heuristik.

K otestování nově představené metriky nejdříve v experimentech přepočítáváme výsledky mezinárodní výzvy *CHeSC2011* a zkoumáme chování algoritmů z obou frameworků. Zajímavým výstupem z provedených pozorování je kvalita řešení samotných stavebních bloků hyper-heuristik ve frameworku *SEAGE*, tedy metaheuristik, nad jednotlivými instancemi domény problému SAT.

V závěru práce shrnujeme veškeré získané znalosti a na základě výsledků experimentů zhodnocujeme framework *SEAGE* z pohledu aktuálního stavu výzkumu hyper-heuristik.

Klíčová slova SAT, TSP, metaheuristika, hyper-heuristika, implementace hyper-heuristiky, implementace evaluátoru, ohodnocení výsledků heuristik, interpretace řešení, metrika, SEAGE, HyFlex, CHeSC2011, ISEA, AdapHH, EPH, PHUNTER

Abstract

This thesis is focused on the evaluation of the optimization framework *SEAGE* in the context of the current research state in the hyper-heuristic field. This is done using our new metric, which evaluates algorithms objectively according to the quality of the optimization problem instances.

The comparison is based on comparing already implemented heuristics in the *SEAGE* and hyper-heuristics created for the international competition *CHeSC2011*.

At the beginning of this thesis, we describe our motivation, goals and also the current state of hyper-heuristic research thanks to which we found out about the *HyFlex* framework together with the hyper-heuristics of the *CHeSC2011* participants.

During the implementation we encountered a series of problems that discouraged us from using the already implemented evaluator and metric in the *HyFlex* framework. We describe these problems in detail and present a new metric and evaluator, which solves mentioned problems and also brings a certain form of improvement into the *SEAGE* framework. For the introduction of hyper-heuristic creation in the *SEAGE* framework we've created one that uses the newly introduced improvement.

For the testing of the newly introduced metric we first recalculate the results of the international competition *CHeSC2011* and examine the behavior of algorithms from both the frameworks. An interesting output has emerged from these observations the quality of the hyper-heuristic building blocks (metaheuristics) solutions in the *SEAGE* framework for the problem domain SAT is really good.

Finally, we summarize all the gained observations and based on the results from experiments we evaluate the *SEAGE* framework in the terms of the current state of hyper-heuristic research.

Keywords SAT, TSP, metaheuristic, hyper-heuristic, implementation of hyper-heuristic, implementation of evaluator, evaluation of heuristic, solution interpretation, metric, SEAGE, HyFlex, CHeSC2011, ISEA, AdapHH, EPH, PHUNTER

Shrnutí

Motivace

Naší motivací k napsání této práce je přiblížení a rozšíření optimalizačního frameworku *SEAGE* o evaluátor s metrikou, jež poskytne objektivní ohodnocení kvality řešení metaheuristik a hyper-heuristik.

Cíl práce

Hlavním cílem této bakalářské práce je návrh a implementace propojení evaluátoru hyper-heuristik *HyFlex* a optimalizačního frameworku *SEAGE* a zhodnocení optimalizačního frameworku *SEAGE* z pohledu aktuálního stavu výzkumu hyper-heuristik.

Mezi podcíle patří následující. Seznámení se s problematikou hyper-heuristik a zpracování rešerše aktuálního stavu výzkumu v této oblasti. Seznámení se s frameworkem *SEAGE* a *HyFlex*. Navržení a implementování způsobu propojení frameworku *SEAGE* s frameworkem *HyFlex* a následná realizace experimentů pro evaluaci metaheuristik a hyper-heuristik. Implementace a otestování vylepšení hyper-heuristiky.

Postup

Na počátku práce jsme shromáždili hyper-heuristiky účastníků mezinárodní výzvy *CHeSC2011*, které byly vytvořeny v prostředí *HyFlex*. Po realizaci prostředí *HyFlex* na našich systémech jsme ověřili jejich věrohodnost v sérii experimentů a porovnali s referenčním řešením z webových stránek výzvy. K objektivnímu porovnání těchto hyper-heuristik s těmi ve frameworku *SEAGE* jsme vytvořili novou Unit metriku. Pomocí ní provedli sérii experimentů a na základě výsledků kriticky zhodnotily framework *SEAGE*. V závěru práce jsme pro představení snadnosti vytvoření nových hyper-heuristik v prostředí *SEAGE* vytvořili takovou, která využívá nově implementované vylepšení objektivního ohodnocení kvality řešení heuristik.

Výsledky práce

V rámci práce jsme úspěšně navrhli a implementovali novou metriku společně s evaluátorem, který ji využívá. Díky návrhu tohoto evaluátoru nebyl problém jej zahrnout do frameworku *SEAGE* ani do upraveného prostředí využívající *HyFlex* pro spuštění hyper-heuristik. S využitím ohodnocení za pomoci nové Unit metriky jsme provedli sérii testů a připravili si tak data pro finální zhodnocení frameworku *SEAGE* z pohledu aktuálního stavu výzkumu v oblasti hyper-heuristik.

Závěr

Z dat získaných experimentů je patrné, že samotné metaheuristiky, které ve frameworku *SEAGE* slouží jakožto stavební bloky pro tvorbu hyper-heuristik, nedosahují podobných výsledků jako hyper-heuristiky z výzvy *CHeSC2011*. Když ale srovnáme jejich výsledky jednotlivých instancí optimalizačních problémů, tak pro doménu problému SAT jsou s nimi občas srovnatelné. To platí i pro nově představenou hyper-heuristiku, jejíž implementace je pouze na desítkách řádků kódu. Framework *SEAGE* má tedy veliký potenciál v oblasti hyper-heuristik, díky využití již samotně velmi výkonných metaheuristik.

Seznam zkratek

ACO	Ant Colony Optimalization
AdapHH	An Adaptive Hyper-heuristic
CHeSC2011	Cross-Domain Heuristic Search Challange 2011
EPH	Evolutionary Programming Hyper-heuristic
GA	Genetic Algorithm
HyFlex	Hyper-heuristics Flexible Framework
ISEA	Iterated Search Driven by Evolutionary Algorithm
PHUNTER	Pearl Hunter
SA	Simulated Annealing
SAT	Boolean Satisfiability Problem
SEAGE	Search Agents Framework
TS	Tabu Search
TSP	Travelling Salesman Problem

Kapitola 1
Úvod

1.1 Motivace

Každý z nás je vybaven komplexním orgánem, mozkem, využívající heuristik (Sekce 2.2) k řešení obrovského množství optimalizačních problémů (Sekce 2.1). I přesto je však výzkum heuristik poměrně novým vědním oborem, který od představení prvních formálních studií heuristik, mezi lety 1940 – 1980, urazil dalekou cestu. [1]

Dlouhou dobu mělo studium heuristik potíže s tím, že nebylo bráno vážně. Objevoval se názor, že oproti algoritmům postrádají jakousi analytickou čistotu zaslouženou akademickým výzkumem. Nicméně se dá těžko argumentovat proti rozsahu jejich přínosu u řešení optimalizačních problémů [2], kterým potvrzují svou aktuálnost a užitečnost pro další zkoumání.

Naší motivací k napsání této práce je přiblížení a rozšíření optimalizačního frameworku *SEAGE* (Sekce 2.5.1) o evaluátor (Sekce 2.6) s metrikou (Sekce 2.7), jenž poskytne objektivní ohodnocení kvality řešení metaheuristik (Sekce 2.3) a hyper-heuristik (Sekce 2.4).

1.2 Cíl práce

Hlavním cílem práce je zhodnocení optimalizačního frameworku *SEAGE* z pohledu aktuálního stavu výzkumu v oblasti hyper-heuristik využitím nově představené metriky umožňující snadnou porovnatelnost metaheuristik a hyper-heuristik s konkurencí.

1.3 Dosažení cíle

Hlavního cíle práce chceme dosáhnout srovnáním nejlepších hyper-heuristik účastníků mezinárodní výzvy *CHeSC2011* s heuristikami ve frameworku *SEAGE* nad doménami problémů TSP a SAT (jiné problémy v *SEAGE* implementované nejsou).

Aby bylo možné vyvozovat závěry, je zapotřebí získat data, u kterých je možná ověřitelnost jejich reprodukovatelnosti. Daty je v tomto kontextu myšlena množina řešení optimalizačních problémů (Sekce 2.1), získaných vybranými metaheuristikami a hyper-heuristikami.

Reprodukovatelnost v práci zajišťujeme vlastním rozšířením implementace frameworku *HyFlex*, tak, aby odpovídal podobě, v jaké byl použit v *CHeSC2011*. Webové stránky výzvy nabízejí referenční řešení jednotlivých účastníků nad množinou použitých instancí problémů (dále jen „instance“). V této práci tyto data reprodukujeme na našem systému a porovnáváme s referenčními pro ověření jejich podobnosti.

Získaná data se ale nedají sama o sobě využít k vyvozování závěrů. K tomu je zapotřebí jejich ohodnocení takovým způsobem, aby se zajistilo objektivní zhodnocení informace v nich obsažené. Hodnota musí umožnit porovnání dat získaných jednotlivými heuristikami mezi sebou.

Pro tyto účely v práci představujeme dvojici evaluátorů. První je již implementovaný ve frameworku *HyFlex* a využívá bodový systém Formule 1. Jeho nevýhodou je, že je téměř nicneříkající. Slouží hlavně k rozřazení hyper-heuristik a metaheuristik na základě jejich řešení série předdefinovaných instancí. Nedává žádnou informaci o kolik je jedna hyper-heuristika či metaheuristika lepší než jiná. Z tohoto důvodu v práci představujeme nový evaluátor s námi nově navrženou Unit metrikou.

Za účelem demonstrace práce s frameworkem *SEAGE* v něm vytváříme novou hyper-heuristiku využívající nově navrhnuté a implementované vylepšení.

Na závěr provedeme sérii experimentů za použití nově implementovaného evaluátoru. Ze získaných poznatků zhodnotím optimalizační framework *SEAGE* z pohledu aktuálního výzkumu v oblasti hyper-heuristik.

1.4 Výzkumný charakter práce

Tato práce je základem pro vědecký článek, ve kterém plánujeme v rámci výzkumného programu *VýLeT 2021* uplatňovat získané výsledky a navazovat na námi formulované otázky problému objektivní formulace kvality algoritmů nad optimalizačními problémy vylepšením a rozšířením použitelnosti námi představené Unit metriky na další domény problémů. Nově představená implementace našeho evaluátoru algoritmů ve frameworku *SEAGE* a *HyFlex* neslouží pro komerční sféru, ale vytváří základ pro další navazující výzkum aktuálnosti frameworku *SEAGE* v oblasti hyper-heuristik.

1.5 Aktuální stav výzkumu hyper-heuristik

Termín hyper-heuristika je poměrně mladý. Jeho definice vznikla kolem roku 2000 a byla chápána jako *„heuristika vybírající si heuristiky“*. Nicméně myšlenku vytváření vysoko-úrovňové heuristiky lze datovat do první poloviny šedesátých let minulého století. [3]

Během posledních několika let se objevila řada prací, odborných výzkumů a článků na téma hyper-heuristik. V následující sekci zmíníme několik z nich.

V roce 2011 se uskutečnila mezinárodní výzva *CHeSC2011* [4], během níž měli účastníci možnost porovnat své hyper-heuristiky mezi sebou. Do této výzvy se zapojily desítky zájemců z celého světa, nevyjímaje ani Českou republikou (Sekce 2.4.4). Hyper-heuristiky se vyvíjely ve frameworku *HyFlex* [5], který byl pro tuto výzvu vytvořen. Jednou z jeho hlavních výhod je, že implementuje veškeré komponenty pracující s optimalizačními problémy, jako jsou například reprezentace řešení a nízko-úrovňové heuristiky (Sekce 2.2). Výzkumníci se tedy mohou zaměřit pouze na implementaci hyper-heuristik. Z této soutěže vzešlo mnoho prací, které ovlivnily další výzkum v této oblasti. Jednou z nich je i navazující práce *LeanGIHH* [6] (Sekce 2.4.5) nad hyper-heuristikou výherce této výzvy *AdapHH* [7] (Sekce 2.4.3), vytvořená v roce 2016, která je její zjednodušenou variantou. Se zachováním kvality řešení snižuje původních 2 324 řádků kódu na pouhých 288.

V roce 2013 vyšel článek *Hyper-heuristics: a survey of the state of the art* [8] shrnující veškerou vědeckou literaturu do roku 2012, historii, trendy a jakým směrem se bude dál vyvíjet oblast hyper-heuristik. Autoři v něm představili rozdělení hyper-heuristik do dvou hlavních kategorií (Sekce 2.4.1). Do první z nich se řadí takové, které selektují heuristiky. Do druhé pak ty, jež heuristiky generují. Záměrem autorů bylo povzbuzení větší spolupráce podobně smýšlejících komunit, převážně těch, které se zaměřují metaheuristikami a strojovým učením.

1.5. Aktuální stav výzkumu hyper-heuristik

S návrhem tří užitečných způsobů pro generování heuristik hyper-heuristikami přichází v roce 2016 článek *Automated design of production scheduling heuristics: A Review* [9]. Těmito návrhy jsou nová reprezentace kandidátních heuristik, které definují vyhledávací prostor, optimalizační algoritmy k prohledávání vyhledávacího prostoru a fitness funkce, jež je využita k zjištění kvality kandidátních heuristik. Autoři taktéž v článku rozřazují hyper-heuristiky do dvou kategorií podle použitých učících metod na supervizované a nesupervizované (Sekce 2.4.1). [9]

Jeden z nejaktuálnějších článků z roku 2019, *Recent advances in selection hyper-heuristic* [3], se primárně zaměřuje na selektivní hyper-heuristiky. Představuje kritickou diskuzi, nejnovější trendy a směr, kterým se bude výzkum dále ubírat. Mimo jiné článek do hloubky probírá již zmíněný framework *HyFlex* a mezinárodní výzvu *CHeSC2011*. Popisuje, jak tato výzva probíhala, jakým způsobem se přerozdělovaly body, její přínos v oblasti selektivních hyper-heuristik a práce ovlivněné právě frameworkem *HyFlex*.

Kapitola 2
Teorie

2.1 Optimalizační problémy

Optimalizační problémy spočívají v hledání nejlepšího řešení z množiny všech dostupných. Lze je rozdělit do dvou základních kategorií z hlediska parametrů. [10]

- Diskrétní
  - Hledané hodnoty proměnných při diskrétní optimalizaci musí patřit do diskrétní množiny.
- Spojité
  - Hledané hodnoty proměnných při spojité optimalizaci mohou být z určitého spojitého intervalu.

2.1.1 Problém splnitelnosti booleovské formule (SAT)

Problém splnitelnosti (SAT) je úlohou matematické logiky, snažící se zjistit, jestli mohou být proměnné daného booleovského výrazu s proměnnými nahrazeny hodnotami pravda nebo nepravda tak, aby byl výsledný výraz (formule) splnitelný. Pokud tohoto cíle nedosáhneme a pro všechna možná přiřazení má výraz hodnotu nepravda, pak označujeme formuli za nesplnitelnou.

Problém splnitelnosti (SAT) je problémem rozhodování o splnitelnosti formule. Problém maximální splnitelnosti je variací tohoto problému, u které se snažíme maximalizovat počet klauzulí formule v konjunktivní normální formě. Tedy ve formě konjunkcí jedné nebo více klauzulí, kde je klauzule disjunkcí literátů. [11]

Následující výraz reprezentuje formuli problému SAT v konjunktivní normální formě.

$$formula = (x1 \vee x2 \vee x4) \wedge (x1 \vee x3) \wedge (x2 \vee x4)$$

2.1.2 Problém obchodního cestujícího (TSP)

Problém obchodního cestujícího (TSP) je diskrétním optimalizačním problémem, který matematicky vyjadřuje a zobecňuje úlohu nalezení nejkratší možné cesty procházející všemi zadanými body (městy) na mapě. Hledá se tedy taková cesta, kterou se navštíví všechna města právě jednou s ohledem na minimalizaci celkové vzdálenosti nebo ceny, kterou mohou být spojení mezi jednotlivými městy ohodnocena a skončí ve výchozím městě. I přes poměrně snadnou formulaci se tento problém řadí do množiny NP-těžkých problémů a není jej možné hrubou silou rychle vyřešit, neboť s rostoucím počtem měst velice rychle narůstá počet všech možných cest. [12]

2.2 Heuristiky

Algoritmy používané pro řešení optimalizačních problémů můžeme také dělit z hlediska přístupu k jejich řešení na deterministické a heuristické. Deterministické algoritmy nacházejí přesné optimální řešení, ovšem s rostoucím stavovým prostorem roste i jejich časová náročnost. Heuristické algoritmy se také snaží nalézt optimální řešení, ale berou ohled na časovou náročnost. Jsou navrženy k řešení problémů rychlejší a daleko efektivnější cestou, omezením se na podmnožinu stavového prostoru, snížením přesnosti a preciznosti. [13]

2.2.1 Náhodné prohledávání

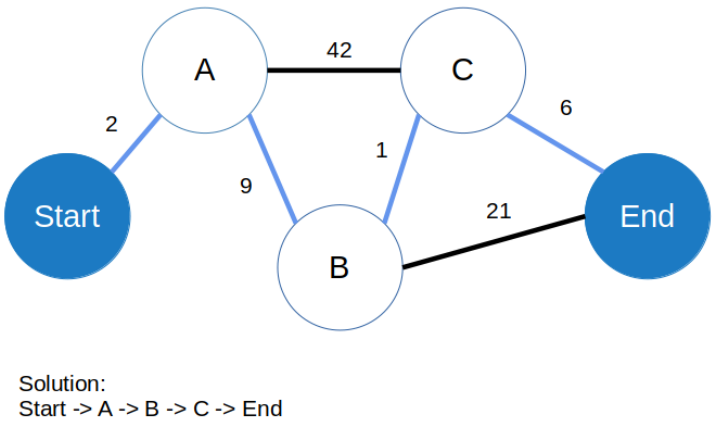
Nejjednodušší heuristikou je náhodné prohledávání. Namísto výpočetně složitého řešení optimalizačních problémů aplikuje matematickou metodu „pokus-omyl“, během níž nijak nevyužívá znalostí o řešeném problému a provádí opakovaný náhodný výběr řešení z množiny všech možných. Jednou z výhod tohoto prohledávání je možnost snadného paralelního běhu, a tedy získání daleko většího množství řešení než jiné heuristiky. [14]

2.2.2 Hladový algoritmus

Hlavní myšlenka hladového algoritmu není nijak složitá, v každém kroku výpočtu volí lepší řešení, než je to aktuální, s vidinou postupného získání globálního optima. Tento postup opakuje, dokud nachází lepší řešení. Lokální rozhodování však mohou vést k lokálním optimům, v nichž algoritmus uvízne. [15]

Volba počátečního bodu rozhoduje, zdali se algoritmus dostane do blízkého lokálního optima. Proto se často používá modifikace tohoto algoritmu, při které je opakovaně spouštěn z náhodných počátečních bodů. Výsledné řešení je to nejlepší z nich.

Na následujícím obrázku je ukázka aplikace hladového algoritmu při hledání cesty grafem s ohodnocenými hranami tak, aby jejich součet byl nejmenší možný.



Obrázek 2.1 Ukázka hladového algoritmu

2.3 Metaheuristiky

Jak již předpona „*meta*“ pocházející z řečtiny napovídá, jedná se o algoritmy vyšší obecnější úrovně, které řeší řadu problémů klasických heuristik. Jedním z hlavních je například už zmíněné uvíznutí v lokálních optimech stavového prostoru. Metaheuristiky ladí a využívají spolupráce nižších heuristik a tím dosahují daleko lepších výsledků, než jakých by mohly dosáhnout nižší heuristiky samy.[16]

2.3.1 Klasifikace

Jedním přístupem k charakterizaci typů vyhledávacích strategií je jejich rozdělení na lokální a globální vyhledávání.

- Lokální vyhledávání
  - Snaha o zlepšení jednoduchých lokálních vyhledávacích algoritmů. Jedním z nich je například již zmíněný hladový algoritmus, který nezaručuje nalezení globálního optimálního řešení.
- Globální vyhledávání
  - Mnoho metaheuristik bylo navrženo tak, aby problém s lokálním optimem řešily. Do této skupiny se řadí například simulované žíhání a tabu vyhledávání, které může být bráno také jako metaheuristika založená na lokálním i globálním vyhledávání.
  - Druhou skupinou jsou populační algoritmy, jako je například optimalizace mravenčích kolonií a genetický algoritmus. Namísto modifikace a vylepšování pouze jednoho řešení se snaží pracovat s celou skupinou kandidátních řešení (populací).

2.3.2 Genetický algoritmus

Genetický algoritmus je náhodný vyhledávací algoritmus, jenž byl vytvořen za účelem imitace mechaniky přírodní selekce a genetiky. Operuje s biologicky podobnými strukturami, které se vyvíjí v čase podle pravidla přeživší nejsilnějšího použitím náhodné, ale strukturované výměny informace. [17] Každá nová generace (populace) vychází z té původní, ze které využívá ty nejsilnější členy. Genetický algoritmus nepracuje s jednotlivými parametry (geny), ale s jejich zakódovanou strukturou (chromozom).

2.3.3 Tabu prohledávání

Jedná se o metaheuristiku využívající metody lokálního prohledávání k řešení optimalizačních problémů. Problémy s možným uvíznutí v lokálních optimech se snaží řešit umožněním podstoupení zhoršujících kroků, pokud nelze podstoupit žádné zlepšující. Jako další řešení tohoto problému představuje restrikce (tabu) k zabránění vracení se k již navštívených řešení. Prohledávání takto iteruje, dokud není splněna určitá podmínka (maximální počet iterací nebo dosažení prahového skóre). [18]

2.3.4 Simulované žíhání

Simulované žíhání vychází z techniky, při které se opakovaným zahříváním a ochlazováním materiálu (žíhání) zlepšuje jeho kvalita.

Na počátku simulovaného žíhání se nastaví proměnná představující teplotu na vysokou hodnotu. Během procesu prohledávání dochází k pomyslnému ochlazovaní teploty, snižováním její

hodnoty. Čím vyšší teplota je, tím je umožněna častější volba zhoršujících řešení oproti aktuálnímu. Toto řeší problém s lokálními optimy, ze kterých metaheuristika ze začátku snáze uniká. Se snižováním teploty se snižuje i šance přijmutí zhoršujícího řešení, a tak dochází k postupnému umožnění prohledávání cílového prostoru, které je potenciálně blízké globálnímu optimu. Proces postupného ochlazování je to, co tuto metaheuristiku činí velice efektivní při hledání řešení blízké optimálnímu a při řešení velkých problémů obsahující četný počet lokálních optim. [19]

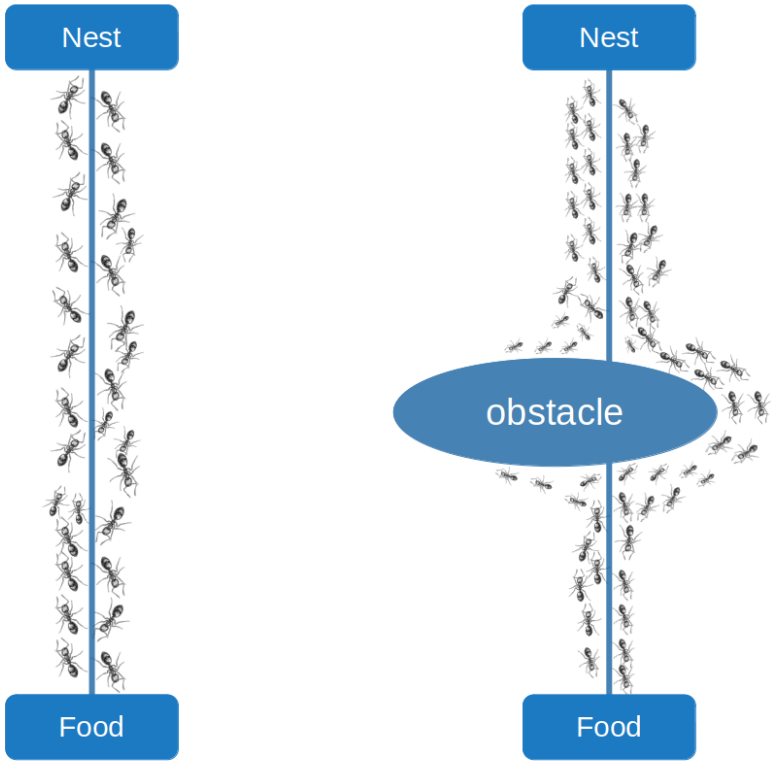
2.3.5 Mravenčí kolonie

Optimalizace mravenčí kolonií je rozšířením tradičních konstruktivních heuristik o schopnost využití zkušeností z průběhů předešlých řešení problémů. Jak je z názvu patrné, tak se vychází z pozorování chování mravenčích kolonií, a to při shánění a sběru potravy. Jejich schopnost se dostat od zdroje potravy zpátky do mraveniště je zajištěna prostřednictvím feromonů. Při hledání potravy je mravenci ukládají po cestě na zem. Cesta s největší koncentrací této látky je upřednostňována před ostatními. Dodržováním tohoto postupu celou kolonií může docházet ke vzniku nejkratší cesty.

Potenciální problém předčasné konvergence k suboptimálnímu řešení se snaží řešit odpařování (zapomínání) feromonu.

V této metaheuristice se mravenci nacházejí v diskrétním světě, lze řídit jejich vnitřní stavy a modifikovat množství a dobu odpařování zanechaného feromonu. [20]

Na následujícím obrázku je ukázka chování mravenců při hledání potravy před a po přerušení jejich původní cesty. Díky pokládání nové stopy feromonů se kratší cesta okolo překážky zvolí jako nová nejkratší.



Obrázek 2.2 Ukázka chování mravenčí kolonie

2.4 Hyper-heuristiky

Hyper-heuristiky jsou heuristické prohledávací metody snažící se automatizovat proces výběru, kombinování, generování a adaptování jednodušších heuristik pro efektivní řešení optimalizačních problémů. Myšlenka za tímto přístupem je taková, že každá heuristika má své silné i slabé stránky nad určitými typy problémů. Tam kde jedna pokulhává, může jiná excelovat, a jejich kombinováním se tyto ztráty snaží kompenzovat. Pro vstupní instanci se tedy hyper-heuristika rozhodne jak a jaké heuristiky by se na ní aplikovat.

Oproti metaheuristikám, jež prohledávají prostor řešení problémů, se hyper-heuristiky zaměřují na prohledávání prostoru heuristik. V dané situaci se tedy snaží o nalezení nejlepší heuristiky, nebo jejich posloupnosti pro danou situaci namísto snahy o přímé řešení problému. Hledá se obecně použitá metodologie namísto řešení instance. [21]

2.4.1 Klasifikace

Hyper-heuristiky lze rozřadit do dvou základních kategorií, kde každá z nich volí jiný přístup při řešení optimalizačních problémů.

První z nich lze popsat jako *„Heuristika vybírající si heuristiky“* a znamená to, že framework implementující hyper-heuristiku obsahuje množinu před-vytvořených heuristik, známé pro řešení určitých problémů. Úkolem procesu prohledávání je nalézt nejlepší možnou posloupnost aplikací těchto heuristik pro efektivní řešení problému. Během výpočtu se v jednotlivých krocích provádí izolované výpočty vybranými heuristikami, řešení nejlepší z nich se předá dalšímu kroku a ten s ním dál pracuje.

Druhou lze popsat jako *„Heuristika vytvářející heuristiky“* a toto řeší využitím komponentů známých heuristik. Frameworku jsou poskytnuty stavební kameny těchto známých heuristik. [22]

2.4.2 EPH

EPH (Evolutionary Programming Hyper-heuristic) hyper-heuristika byla vytvořena Davidem Meignanem v mezinárodní výzvě *CHeSC2011*, na které obsadila páté místo. Vychází z přístupu evolučního programovaní a ko-evoluce. Během níž je množina řešení (populace) vyvíjena aplikací sekvence heuristik, jež jsou popořadě vyvíjeny dle jejich výkonu separátním evolučním algoritmem. [23]

2.4.3 GIHH/AdapHH

Adaptivní hyper-heuristika AdapHH byla vytvořena Mustafou Misirem v mezinárodní výzvě *CHeSC2011*, na které obsadila první místo. Využívá adaptivní dynamickou množinu heuristik (ADHS), jež si zaznamenává a drží informace o výkonu jednotlivých heuristik a odděluje všechny, až na ty nejlepší po určitém počtu iterací. Výkonnostní metrika je založena na jednoduchém indikátoru kvality, jako je schopnost zlepšení a rychlost. Metrika je využita při procesu vyřazování heuristik. [24]

2.4.4 ISEA

ISEA hyper-heuristika byla vytvořena Jiřím Kubalíkem v mezinárodní výzvě *CHeSC2011*, na které obsadila osmé místo. Zakládá se na evolučně iterativním lokálním prohledávání algoritmu POEMS. POEMS je optimalizační algoritmus operující na jednoprvkovém řešení (prototype) a pokouší se jej vylepšit během iteračního procesu. V každé iteraci je spuštěn evoluční algoritmus vyhledávací nejlepší modifikaci prototype. Modifikace je reprezentovaná fixní délkou sekvence

základních akcí ovlivňující kvalitu prototype. Po dokončení evolučního algoritmu se nejlepší vyvinutá sekvence zkontroluje, zdali zhoršuje nebo zlepšuje aktuální prototyp. Pokud jej modifikace nezhoršuje, pak je brána jako nový prototyp pro další iteraci, jinak je zachován původní. Tento proces se opakuje, dokud se nedovrší určitého počtu iterací. [25]

2.4.5 LeanGIHH

Lean-GIHH hyper-heuristika byla vytvořena jakožto zjednodušená varianta hyper-heuristiky GIHH (AdapHH). Za použití středních hodnot Accidental Complexity Analysis (ACA) a techniky pro redukci algoritmické komplexnosti bez ztráty výkonu se Lean-GIHH podařilo snížit původních 2 324 řádek kódu na pouhých 288. [6]

2.4.6 PHUNTER

Tato hyper-heuristika byla vytvořena Fan Xueem v mezinárodní výzvě *CHeSC2011*, na které obsadila čtvrté místo.

Lov perel je tradiční způsob potápění se za účelem získání perel ústřic nebo pro lov jiných mořských tvorů. Během tohoto procesu se lovci musí opakovaně potápět a prohledávat dno v několikametrové hloubce.

PHUNTER hyper-heuristika se lovem perel inspiruje. Seskupuje, testuje, řadí, vybírá a kombinuje heuristiky nižších úrovní pro vytvoření imitace potápěče. Heuristiky zajišťující lokální prohledávání lze chápat jako proces potopení a dělí se do dvou kategorií. První z nich je šnorchlování, během kterého se provádí série lokálních prohledávání a zastavují se při nalezení zlepšení. Druhá kategorie je hloubkové potopení využívající nejlepší nalezená řešení při šnorchlování, nad nimiž provádí prohledávání, dokud nachází zlepšení. [26]

2.5 Optimalizační frameworky

Optimalizačním frameworkem lze chápat prostředí implementující sadu softwarových nástrojů pro zajištění správné a znovupoužitelné implementace heuristik.

2.5.1 SEAGE

Optimalizační framework *SEAGE* byl vytvořen v rámci disertační práce Ing. Richardem Málkem na Fakultě elektrotechnické ČVUT v Praze.

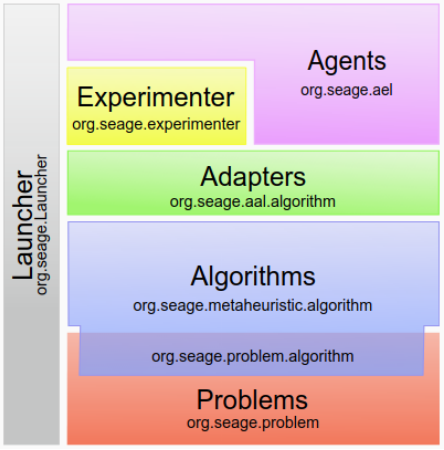
Jednou ze základních misí tohoto projektu je nabídnout knihovnu optimalizačních problémů a implementací algoritmů. Většina aktuálně implementovaných problémů je diskrétní povahy. Každý z implementovaných algoritmů může být rovnou aplikován na nový optimalizační problém bez nutnosti využití žádného jiného rozšíření frameworku.

Na následujícím obrázku je zachycena aktuální podoba pokrytí optimalizačních problémů jednotlivými algoritmy. Stav je buď hotový (/), skoro hotový (||), problém (!) nebo zatím neimplementováno (–). [27]

	GRASP	Genetic Algorithm	Tabu Search	Simulated Annealing	Ant Colony	Particle Swarm	Firefly Algorithm
Traveling Salesman	⏸	✓	✓	✓	✓	⏸	⏸
Satisfiability	⏸	✓	✓	✓	✓	–	–
Jobshop Sheduling	–	✓	✓	⏸	!	–	–
Quadratic Assignment	–	–	⏸	⏸	–	⏸	⏸

Obrázek 2.3 Problémy a algoritmy ve frameworku SEAGE

Architektura frameworku *SEAGE* je rozdělena do několika vrstev, které si vyměňují informace a data směrem zespodu nahoru. Každá vrstva nemá znalost o vrstvách nad ní, pro získání potřebných dat využívá pouze tu přímo pod ní. Na následujícím obrázku je zachycena architektura. [27]



Obrázek 2.4 Architektura frameworku SEAGE

2.5.2 HyFlex – CHeSC2011

HyFlex je framework navržený k umožnění vývoje, testování a porovnání iterativních hyper-heuristik. Pro zajištění těchto cílů je využita modularita a koncept rozložení heuristického vyhledávacího algoritmu na dvě části. Jednu specializovanou na algoritmy a hyper-heuristiky a druhou problémově závislou, která je frameworkem *HyFlex* implementovaná. Jedná se o pomyslnou doménovou bariéru mezi problémově specifickými heuristikami a hyper-heuristikami. *HyFlex* rozšiřuje tento koncept udržováním populace na úrovni domény problému.

Framework je napsán v jazyce Java, který je běžně využíván mnoha výzkumníky. Dalším přínosem je i jeho objektivní návrh, nezávislost na platformě a automatizovaná správa paměti. Pohledem z nejvyšší úrovně abstrakce má framework pouze dvě abstraktní třídy, ProblemDomain a HyperHeuristic.

CHeSC2011 byla první mezinárodní Hyper-heuristickou výzvou snažící se spojit odborníky z oblasti operačního výzkumu počítačové vědy a umělé inteligence se zájmem o vývoj hyper-heuristik. Úkolem byla implementace takové vyhledávací strategie, využívající sadu problémově orientovaných heuristik nižší úrovně. Tato sada se lišila pro každou doménu problému. [5]

2.6 Evaluátor

V kontextu této práce je evaluátor chápán jakožto kód starající se o načítání dat, ohodnocení dat za použití metrik a následného uložení výsledku.

2.7 Metrika

Metriky jsou definovány nad matematickou strukturou, pomocí níž se formálním způsobem definuje pojem vzdáleností mezi elementy z množiny X. Metrický prostor M je dvojice (X, ψ), kde X je libovolná neprázdná množina a ψ je metrika zobrazení $\psi : X \times X \longrightarrow R$ splňující následující axiomy pro libovolná $x, y, z \in M$. [28]

- Nezápornost: $\psi(x, y) \geq 0$
- Totožnost: $\psi(x, y) = 0 \Longleftrightarrow x = y$
- Symetrie: $\psi(x, y) = \psi(y, x)$
- Trojúhelníková nerovnost: $\psi(x, z) \leq \psi(x, y) + \psi(x, y)$

Kapitola 3
Evaluace

Evaluátor má za úkol pro každý algoritmus ohodnotit kvalitu jeho řešení optimalizačních úloh s využitím příslušné metriky. V této kapitole popisujeme ty metriky, které v této práci využíváme.

3.1 Bodový systém Formule 1

Bodový systém Formule 1 byl využit v mezinárodní výzvě *CHeSC2011* k vyhodnocení úspěšnosti jednotlivých účastníků. Myšlenka za ním je následující. Nejlepší osmička závodníků dostává během jednotlivých kol závodu 10, 8, 6, 5, 4, 3, 2 a 1 bod dle dosaženého umístění a zbytek nedostane nic. Výsledné skóre závodníka je součtem získaných bodů v jednotlivých kolech. Pokud se tedy koná deset kol, pak je maximální možné skóre 100 bodů.

V případě remízy se body z dosažení daného místa rovnoměrně rozdělí mezi všemi závodníky v remíze, a tím je tedy zachován celkový součet bodů během každého kola. Pokud dojde k remíze i ve finálním skóre, pak se hledí na celkový počet umístění na prvním místě. Pokud je i zde remíza, tak se zohledňuje počet umístění ve druhém místě a tak dále.

3.1.1 Problémy metriky

Problémem není její implementace, ale použití pro ohodnocení kvality řešení jednotlivých hyper-heuristik a metaheuristik. Toto vyplývá z její povahy. Výsledky nedokáže ohodnotit samostatně. Vyžaduje tedy více účastníků, ale zároveň se s jejich různým počtem liší celkové ohodnocení jednotlivců.

3.2 Unit metrika

Námi nově představená Unit metrika operuje nad doménami problémů SAT a TSP. Motivací této metriky je namapování výsledku na interval mezi optimální a snadno dostupné řešení (získané hladovým nebo náhodným algoritmem). Namapovaná hodnota bude určovat kvalitu řešení pro každou ze tří složek – *instanci* (U_i), *problém* ((U_p)) a *experiment* ((U_e)) (metaheuristiku nebo hyper-heuristiku). K řešení problému různorodosti instancí jsou využity metadata. V metadatech je pro každou instanci uložena její velikost, název a hodnota optimálního, náhodného a hladového řešení. Většina z těchto informací je snadno vyhledatelná a tedy známá, co jsme ale museli pro metadata zajistit bylo řešení náhodným a hladovým algoritmem pomocí frameworku *SEAGE*.

3.2.1 Metadata

Metadata jsou dodatečné informace o instancích a uchovávají jejich název, velikost, optimální a snadno dostupné hodnoty řešení (náhodné a hladové). Optimální hodnoty jsou známé a tedy snadno dostupné. Kvalitu náhodného a hladového řešení jsme si dopočítali (medián z 1000 běhů výpočtu).

Pro vygenerování metadat jsme využili framework *SEAGE*, především proto, že již obsahuje implementace problémů TSP a SAT a podporuje snadnou práci s instancemi. Získaná metadata se následně lehce přesunula i do frameworku *HyFlex*.

Výpis kódu 3.1 Ukázka metadat pro instanci z domény problému SAT

```
<Instance greedy="2" id="uf20-0127" optimum="0" random="9" size="20"/>
```

Výpis kódu 3.2 Ukázka metadat pro instanci z domény problému TSP

```
<Instance greedy="1427" id="rat99" optimum="1211" random="8313" size="99"/>
```

3.2.2 Ohodnocení instance

Ohodnocení řešení instance je snadné. Evaluátor z metadat vybere optimální a hladovým algoritmem získané řešení instance. Poté namapuje řešení instance na interval [0,1], kde 0 odpovídá hladovému a 1 teoreticky optimálnímu řešení. Pokud je řešení nedosahuje hladovému řešení, je ohodnoceno nulou. Čím blíže je namapována hodnota jedničce, tím je řešení bližší optimálnímu a tedy lepší.

$$U_i(instance) = map(instance, optimal, greedy) \in [0, 1] \tag{3.1}$$

U řešení optimalizačních úloh je problémové vyčíst, jak moc dobře si daná hyper-heuristika nebo metaheuristika vedla. Pokud začínala s již kvalitním řešením, pak se vypočtené bude pohybovat pouze lehce nad původní kvalitou. Pro vyčtení míry zlepšení se proto zavedlo tzv. delta skóre. To se rovná rozdílu kvality finálního řešení s počátečním.

$$U_i(final\_instance) - U_i(init\_instance) \in [0, 1] \tag{3.2}$$

3.2.3 Ohodnocení domény problému

Jelikož každá instance domény problému je různě náročná, je třeba tuto skutečnost vzít v úvahu při ohodnocování samotné domény. Toto se zařídí pomocí váženého průměru, kde je váhami myšlena velikost instance dostupná v metadatech. Pro ohodnocení domény je tedy třeba množina ohodnocených řešení instancí Unit metrikou (U_i) a jejich váhy (velikosti).

$$U_p(problem) = weighted\_mean(\{all\, U_i(instance\_i)\}, instances\_sizes) \in [0, 1] \tag{3.3}$$

3.2.4 Ohodnocení experimentu

Poslední část metriky, experiment, nevyužívá z metadat žádné informace. Evaluátor provede průměr ohodnocených domén (U_p), které se v experimentu využily, z druhé části metriky (ohodnocení problému). Hodnota experimentu je z intervalu [0, 1] a značí kvalitu algoritmu nad danou množinou instancí.

$$U_e(experiment) = mean(\{all\, U_p(problem)\}) \in [0, 1] \tag{3.4}$$

Kapitola 4
Implementace

Na začátku kapitoly popisujeme, proč jsme nevyužili evaluátor implementovaný ve frameworku *HyFlex*, jaké jsou jeho nevýhody, které nás motivovaly k vytvoření nového. V druhé části kapitoly se věnujeme představení našeho nového evaluátoru společně s jeho implementací v jednotlivých frameworcích. V závěru popisujeme nově implementované vylepšení, které naše metrika do *SEAGE* přinesla a její využití v hyper-heuristikách.

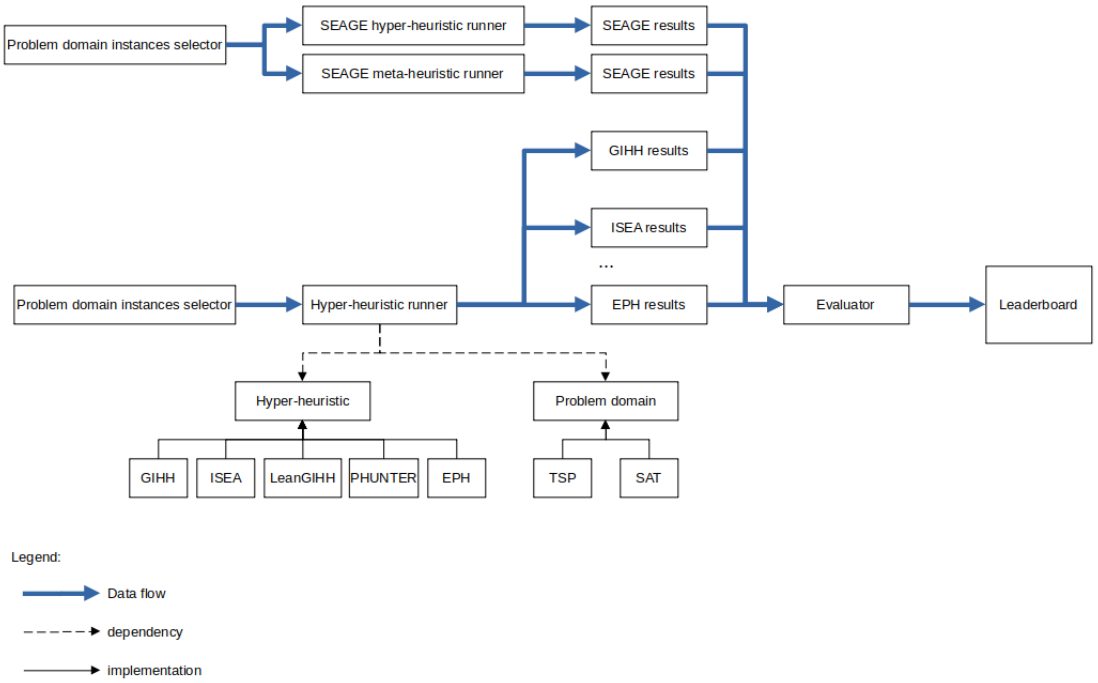
4.1 Prvotní evaluátor

Jak již bylo dříve v textu zmíněno, pro zhodnocení aktuálnosti optimalizačního frameworku *SEAGE* v oblasti hyper-heuristik je třeba množina optimalizačních problémů. Dále je potřeba množina konkurenčních hyper-heuristik, prostředí, ve kterém je lze spustit, a určitý evaluátor, který ze získaných řešení množiny instancí problémů vyhodnotí jejich kvalitu. Tím zároveň poskytne informaci o kvalitě toho, co je vygenerovalo.

Vše zmíněné řeší framework *HyFlex* společně s daty z mezinárodní výzvy *CHeSC2011*. Prvotní myšlenka byla tedy následující: „Proč toho nevyužít?“ Za pomoci zdrojových kódů frameworku *HyFlex* jsme vytvořili přesnou kopii výzvy. Tím se zajistilo prostředí i evaluátor, který je v *HyFlex* už vytvořen. Chyběly jen zdrojové kódy účastníků. Jelikož ne všechny byly dostupné na internetu, tak v této práci využíváme pětici z nich [AdapHH, LeanGIHH, ISEA, EPH, PHUNTER] (Sekce 2.4.1-6). Z domén problémů jsou pro srovnání využity [SAT, TSP] (Sekce 2.1.1-2).

4.1.1 Implementace

V diagramu (Obrázek 4.1) je zachycena naše prvotní implementace prostředí využívající evaluátor ve frameworku *HyFlex*. Ten využívá metriku F1, která má řadu nevýhod. O těch až později (Sekce 4.1.2).



Obrázek 4.1 Diagram našeho prostředí s původním evaluátorem a metrikou F1

V následujících bodech v krátkosti popíšeme, co měly třídy v jednotlivých částech za úkol.

- **Problem domain instance selector**
  - Jedna z našich tříd na vstupu přebírala vstupní parametry od uživatele. Ty ovlivňovaly, jaká část prostředí se spustí.
  - První část se starala o řešení optimalizačních úloh pomocí frameworku *HyFlex*. Pro tuto část se uživatelem definovala doba běhu, opakování výpočtů a která hyper-heuristika nebo metaheuristika se spustí. Výběr instancí problémů jsme definovali v kódu, aby všechny hyper-heuristiky a metaheuristiky řešily stejné optimalizační úlohy.
  - Pro druhou část, evaluátor, se na vstupu nic nedefinovalo. Cesta k adresáři s výsledky předchozího běhu byla definovaná v kódu. Přesně tak, jak to bylo nastaveno ve frameworku *HyFlex*.
- **Hyper-heuristic runner / metaheuristic runner**
  - Za pomoci získaných parametrů předchozí části námi upravená třída frameworku *HyFlex* spustila výpočet definovaných optimalizačních problémů a řešení uložila do souborů s pevnou strukturou.

- **Results**
  - Tato část reprezentuje soubory, ve kterých je uloženo řešení optimalizačních úloh. Soubor má pevně definovanou strukturu. Řádky reprezentují domény problémů a sloupce řešení jednotlivých instancí. Jedná se v určitém pohledu o soutěžní karty, které zachovávají podobu použitou i během výzvy *CHeSC2011*.
- **Evaluator**
  - Vstupní třída měla v kódu nastavenou cestu, ve které se nacházely výsledky předchozího běhu. Tyto data načetla a za pomocí F1 metriky udělila skóre. Aby se výsledek uložil v čitelném formátu, tak jsme do této třídy přidali část kódu ukládající ohodnocená data ve formátu XML.
- **Leaderboard**
  - Tato část reprezentuje získané ohodnocení řešení jednotlivých hyper-heuristik a metaheuristik. Zobrazuje, jak si vedly nad jednotlivými doménami problémů. Formát opět odpovídal tomu, který byl použit ve výzvě *CHeSC2011*.

4.1.2 Problémy evaluátoru

Problémy s tímto řešením se dají rozdělit do dvou větších částí. První je po stránce implementace. Určité třídy přebrané z frameworku *HyFlex* jsou přeplněné kódem. Práce s nimi je tedy obtížnější, než by bylo vhodné. Dle našeho názoru postrádají snadnou srozumitelnost, na kterou touto prací cílíme. Aby kdokoliv, kdo by rád projekt využil a dále rozvíjel nemusel zdlouhavě dumat, jak to vlastně použije. Druhá část byla použitá F1 metrika, která nedokáže výsledky ohodnotit samostatně.

Problémem byla i nutnost získávat data z frameworku *SEAGE*. Ty bylo zapotřebí složitě zapisovat do definovaného souboru a ten přesouvat na požadované místo. Během tohoto procesu tak vznikala možnost lidské chyby.

4.2 Představení nového evaluátoru

Náš nový evaluátor namísto původní metriky využívá nově představenou Unit metriku. Díky jejímu návrhu je umožněno objektivní ohodnocení řešení jednotlivých hyper-heuristik a metaheuristik. Není tedy nutnost evaluátor spouštět nad množinou všech řešení.

Nový návrh evaluátoru umožnil snadnou implementaci v obou frameworcích. Odstraňuje se tak nutnost data mezi nimi složitě přesouvat. Obě prostředí díky němu získají již finální objektivní ohodnocení kvality jednotlivých hyper-heuristik a metaheuristik. Ty pak stačí pouze mezi sebou porovnat.

Aby vše fungovalo, bylo zapotřebí získání metadat pro Unit metriku, vytvoření jádra našeho evaluátoru a metriky, jak pro řešení instancí problémů, tak i pro domény problémů a celkové ohodnocení napříč všemi doménami problémů.

4.3 Evaluátor ve frameworku HyFlex

Náš nový evaluátor ve frameworku *HyFlex* ([29]) zachovává rekonstrukci výzvy *CHeSC2011*. Řešení optimalizačních úloh se taktéž ukládá do souborů s pevnou strukturou. Soubory s řešením jsou pojmenované názvy hyper-heuristik, které je řešily. S každým spuštěním si uživatel může vybrat, jestli chce výsledky uložit do adresáře s automaticky generovaným názvem, nebo do adresáře s názvem dle svého výběru. Po získání dat může uživatel provést výpočet skóre jednotlivých

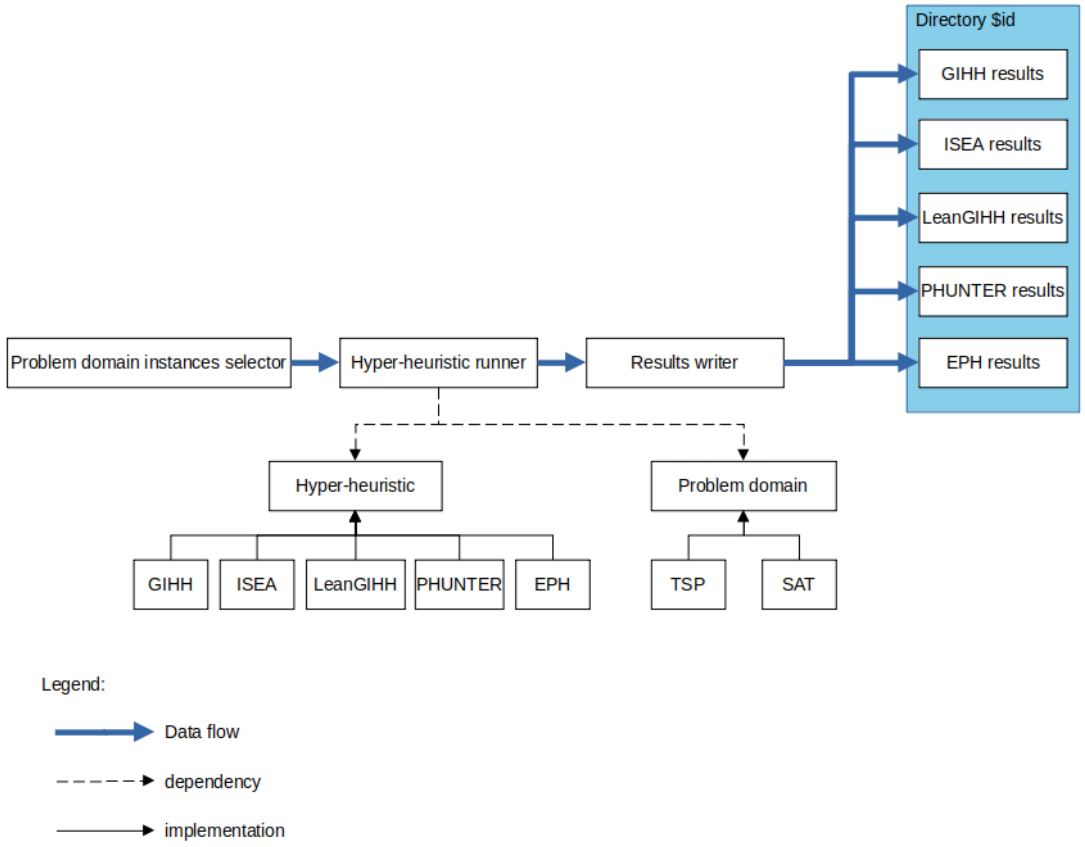
výsledků v zadaném adresáři. Výsledkem je XML soubor s ohodnocením řešení instancí, domén a celkovým.

Výpis kódu 4.1 Ukázka ohodnocení řešení hyper-heuristiky ISEA využitím Unit metriky

```
<algorithm name="ISEA" score="0.8519227908270548">
    <problem avg="0.8246412487011086" name="TSP">
        <instance rat575-hyflex-2="0.964669330210594"/>
        <instance d1291-hyflex-6="0.7753049096137679"/>
        <instance pr299-hyflex-0="0.9997690661120505"/>
        <instance usa13509-hyflex-8="0.8213844416866103"/>
        <instance u2152-hyflex-7="0.8129358841317728"/>
    </problem>
    <problem avg="0.8792043329530009" name="SAT">
        <instance pg-525-2336-hyflex-4="0.9302325581395349"/>
        <instance pg-696-3122-hyflex-5="0.8695652173913043"/>
        <instance hg4-300-1200-hyflex-11="0.7872340425531915"/>
        <instance pg-525-2276-hyflex-3="0.8974358974358975"/>
        <instance jarv-684-2300-hyflex-10="0.8761904761904762"/>
    </problem>
</algorithm>
```

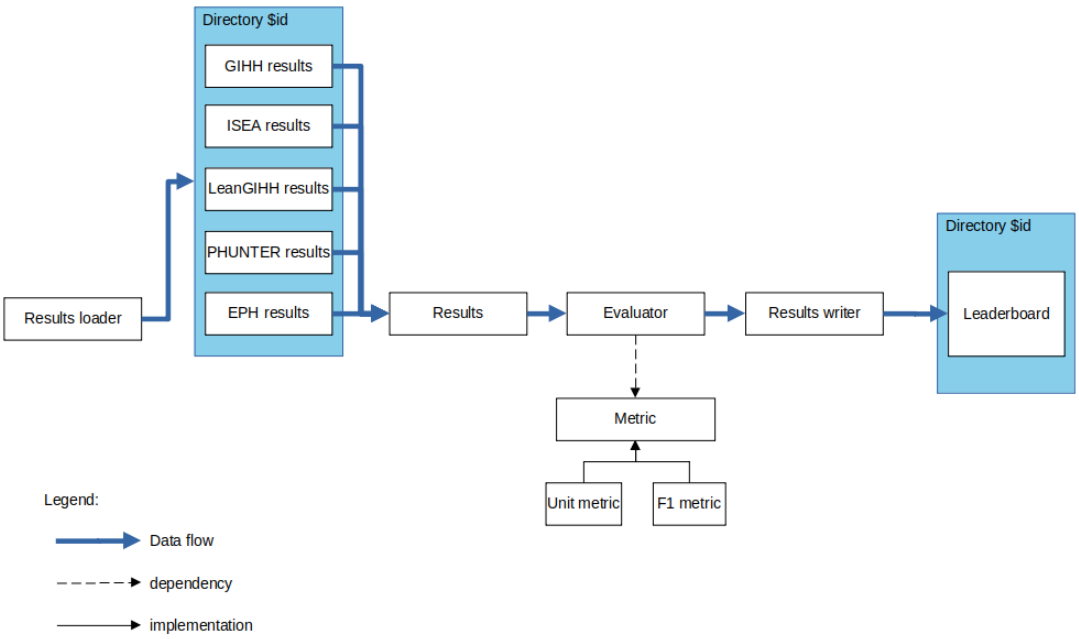
- algorithm
  - Název hyper-heuristiky nebo metaheuristiky a průměr skóre domén.
- problem
  - Název domény problému a vážený průměr instancí.
- instance
  - Název instance a jeho ohodnocení naším novým evaluátorem využívající Unit metriku.

Organizaci jednotlivých logických bloků získávání dat a našeho evaluátoru popíšeme blíže na diagramu (Obrázek 4.2) a (Obrázek 4.3).



Obrázek 4.2 Diagram našeho nového experimentátoru hyper-heuristik ve frameworku HyFlex

- Problem domain instances selector
  - V této části se přebírají od uživatele parametry, kterými se ovlivňuje doba běhu a počet opakování při výpočtu optimalizační úlohy. Dále se přebírá parametr pro určení hyper-heuristiky k použití a adresáře pro uložení souboru s řešením.
- Hyper-heuristic runner
  - V této části probíhá parametry modifikovaný výpočet množiny optimalizačních problémů.
- Results writer
  - Ze získaných dat z předchozích částí se vytvoří soubor s pevnou strukturou uchovaných řešení. Pojmenován je po použité hyper-heuristice. Soubor se uloží do uživatelem definovaného adresáře. Pokud neexistuje, tak se vytvoří.
- Results
  - Představuje samotný soubor s řešením.



Obrázek 4.3 Diagram našeho nového evaluátoru a Unit metriky ve frameworku HyFlex

- Results loader
  - Tato část od uživatele přebírá parametry definující název adresáře s daty a název metriky k použití.
- Results
  - Před dalším zpracováním je potřeba načtená data reprezentovat vnitřní strukturou.
- Evaluator
  - Evaluátor provede ohodnocení výsledku dvěma způsoby. V případě, že uživatel vybral metriku F1, se metrice poskytnou všechny data najednou. V případě Unit metriky se data postupně pošlou k ohodnocení metrikou.
- Results writer
  - Úkolem této části je ohodnocená data uložit do adresáře, ze kterého byla čtena. Pod názvem metriky, která data ohodnocovala.
- Leaderboard
  - Samotný XML soubor s řešením (Kód 4.1).

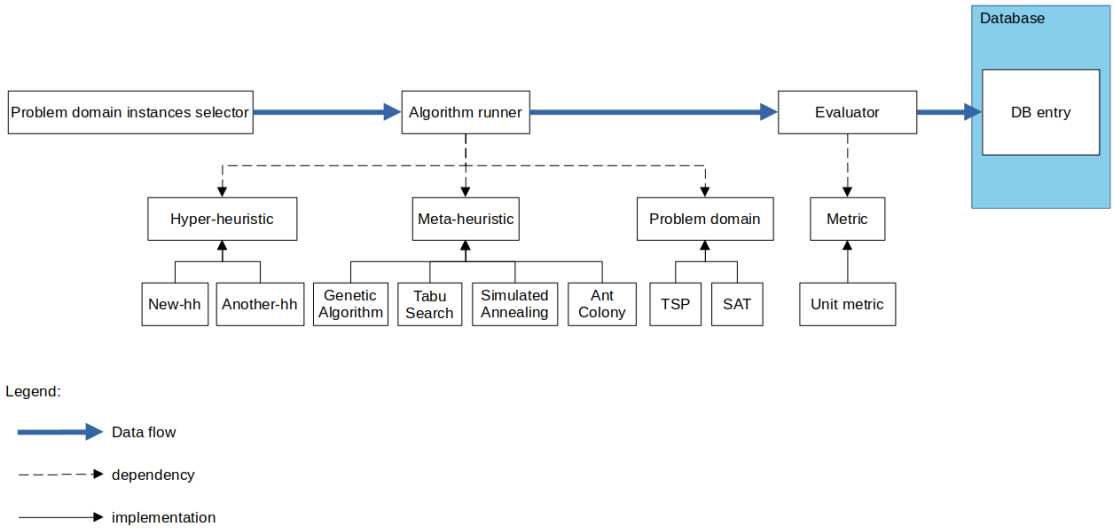
4.4 Evaluátor ve frameworku SEAGE

Díky lehkému konceptu našeho nového evaluátoru nebyl problém jej implementovat i v prostředí *SEAGE* ([30]). Nejdříve bylo zapotřebí upravit několik míst frameworku tak, aby se v něm dal nový evaluátor použít. Oproti *HyFlex* využívá *SEAGE* sofistikovanější způsob ukládání dat, a tím je databáze. Tu bylo třeba upravit tak, aby se do ní mohly zapisovat nové informace o ohodnocení a míře zlepšení řešení.

Liší se také rozdělení výpočtu řešení a následně jeho zpracování evaluátorem, jak tomu je v *HyFlex*. Toto ohodnocení probíhá v každé části výpočtu optimalizačních úloh. Výpočet použitím vybraného způsobu ladění parametrů algoritmu je brán jako experiment a má jedinečný identifikátor. Pod ním se data ukládají do trojice tabulek v databázi.

- solutions
  - Tabulka uchovává jednotlivé řešení instancí.
  - Najdeme zde řešení, počet iterací, datum a skóre řešení.
- experiment_task
  - Tabulka uchovává informace o řešení instance.
  - Najdeme zde název použité instance a domény problému, názvu algoritmu, jeho konfiguraci a statistiku toho, jak si vedl (průměrné řešení, počáteční hodnota, počet iterací atd.), datum počátku výpočtu a jeho konce. V neposlední řadě i skóre a delta skóre získané evaluátorem.
- experiments
  - Tabulka uchovává informace o řešení instancí.
  - Najdeme zde název použitého algoritmu, instancí a domén problémů, konfiguraci danou uživatelem (počet opakování a čas výpočtu), čas počátku výpočtu a jeho konec, celkové skóre algoritmu a JSON se skórem jednotlivých instancí a domén problémů.

V následujícím diagramu (Obrázek 4.4) popisujeme, jak je implementován náš nový evaluátor v prostředí *SEAGE*.



Obrázek 4.4 Diagram využití nového evaluátoru s Unit metrikou ve frameworku SEAGE

- Problem domain instances selector
  - Tato část od uživatele přebírá parametry definující název instance, typ experimentu, čas výpočtu a počet opakování výpočtu.
- Algorithm runner
  - V této části probíhá vlastní výpočet optimalizačních úloh.
- Evaluator
  - Získané výsledky ohodnotí a následně společně s dalšími informacemi uloží do databáze.
- DB entry
  - Reprezentuje data v databázi.

4.4.1 Vylepšení

Dříve bylo v textu zmíněno, že se parametry algoritmů ladí. Tím je myšleno, jakým způsobem se provádí výběr jejich parametrů. U simulovaného žíhání jde například o nejnižší teplotu, nejvyšší teplotu a počet iterací. Ve frameworku *SEAGE* je implementovaná čtveřice těchto ladících přístupů. V této práci představujeme nový vylepšený přístup.

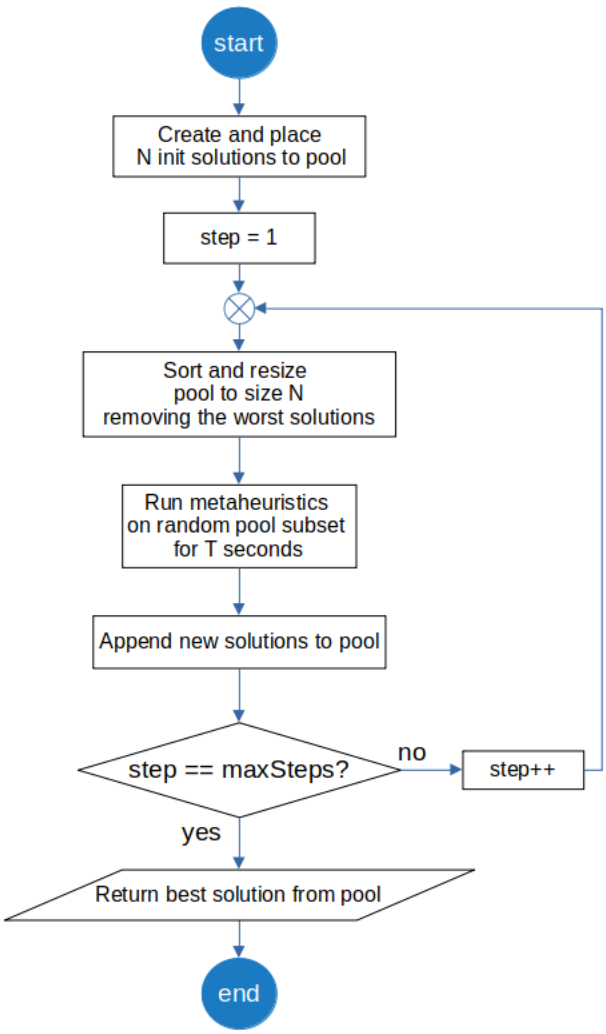
- Default
  - Nastavení parametrů je provedeno využitím jejich nastavené počáteční hodnoty. K té je připočtena, nebo od ní odečtena hodnota závislá na vstupním parametru uživatele.
- Random
  - Nastavení parametrů probíhá náhodným výběrem z intervalu vhodných parametrů.
- Grid
  - Nastavení parametrů probíhá výběrem z omezeného intervalu vhodných parametrů. Omezení je závislé na vstupním parametru uživatele.
- Evolution
  - Nastavení parametrů je dosaženo využitím evolučních algoritmů.
- Feedback
  - Jedná se zatím o koncept nastavení parametrů s využitím již spočtených řešení. Z nich se vybere to nejlepší s ohledem na skóre a delta skóre. Okolí těchto parametrů pak bude využito k získání nových.
  - Základním kamenem pro tento koncept je implementovaný konfigurátor *ExtendedDefaultConfigurator*, který parametry vybírá z ručně nastavených nejlepších z předešlých běhů. Zatím neautomatizovaný přístup slouží k otestování projevu určitého zlepšení při využití jedněch parametrů pro všechny instance dané domény. Při využití nejlepších parametrů jednotlivých instancí z minulých běhů lze očekávat daleko větší zlepšení, ale při ručním nastavováním je toto velmi zdlouhavá práce.

4.5 Implementace hyper-heuristiky

V této sekci se dostáváme k poslednímu podcíli této práce, ve kterém plně využíváme nově implementovanou metriku při měření kvality získaných řešení a při nastavování parametrů metaheuristik vylepšeným konfigurátorem *ExtendedDefaultConfigurator* (Sekce 4.4.1).

Pro představení tvorby hyper-heuristik ve frameworku *SEAGE* volíme hyper-heuristiku takové podoby, jejíž implementace poslouží jako základní kámen pro další navazující práce. Je tedy postavená na velice jednoduché myšlence, a to takové, že při řešení optimalizační úlohy si nejdříve hyper-heuristika předpřipraví pole (pool) inicializačních řešení (fenotypů) stanovené délky a následně nad náhodně vybranou podmnožinou z nich spustí jednotlivé metaheuristiky. Po uplynutí času jejich výpočtu se do pole uloží nově získané řešení a odstraní se ty s nejhorším, tak, aby pole zachovalo stanovenou velikost. Tento proces se opakuje, dokud nevyprší celkový stanovený čas pro výpočet optimalizačního problému. Konečným výsledkem hyper-heuristiky je nejlepší řešení z pole.

Názornější zjednodušený popis funkčnosti hyper-heuristiky je zachycen na následujícím digramu (Obrázek 4.5), na kterém jsou zobrazeny jednotlivé kroky od vytvoření pole inicializačních řešení až po získání nejlepšího z něj.



Obrázek 4.5 Vývojový diagram hyper-heuristiky

4.6 Shrnutí

V této kapitole jsme úspěšně navrhli, popsali a implementovali novou metriku společně s evaluátorem, který ji využívá. Díky návrhu tohoto evaluátoru nebyl problém jej zahrnout do frameworku *SEAGE* ani do našeho upraveného prostředí využívající *HyFlex* pro spouštění hyper-heuristik. Pro představení snadnosti tvorby nových hyper-heuristik v prostředí *SEAGE* jsme vytvořili takovou, která využívá nově implementované vylepšení objektivního ohodnocení kvality řešení heuristik.

Kapitola 5
Experimenty

Experimenty dělíme do dvou částí, ve kterých s využitím námi nově představené Unit metriky přepočítáváme výsledky mezinárodní výzvy *CHeSC2011* získané použitím F1 metrikou a kriticky srovnáváme stavební bloky (metaheuristiky) využité pro tvorbu hyper-heuristik v prostředí *SEAGE* s účastníky této výzvy, které jsme měli možnost zrekonstruovat a otestovat na našem systému. Při porovnávání výsledků mezi algoritmy implantovanými v prostředí *HyFlex* a těch v prostředí *SEAGE* je potřeba brát v úvahu rozdíl v jejich chování při spuštění se stejnými parametry. Kromě zhodnocení samostatných metaheuristik zhodnotíme i novou hyper-heuristiku s konkurencí, abychom zjistili, jak nejjednodušší implementace hyper-heuristiky ve frameworku *SEAGE* obstojí proti konkurenci. Získané pozorování povede ke zhodnocení frameworku *SEAGE* z pohledu aktuálního stavu vývoje hyper-heuristik.

5.1 F1 metrika v experimentech

Na začátku prvních dvou experimentů nejdříve využíváme metriku F1, která byla využita i během mezinárodní výzvy *CHeSC2011*. Ta pro každou instanci seřadí účastníky (algoritmy) dle kvality jejich řešení a na základě dosaženého umístění přidělí body z množiny {10, 8, 6, 5, 4, 3, 2, 1, 0}, algoritmus na prvním místě dostane 10 bodů, na druhém 8 atd. Pokud žádné body už nezbudou, pak je algoritmus na dané instanci hodnocen 0 body a v případě remízy se body daného místa rovnoměrně dělí. Celkové ohodnocení algoritmu je součtem bodů z jednotlivých instancí domén problémů. Při výzvě *CHeSC2011* se využilo pět instancí pro každou z šesti domén problémů, a tedy maximální možný počet získaných bodů byl 300.

5.2 Unit metrika v experimentech

Pro přepočet výsledků v prvních dvou experimentech a ostatních experimentech využíváme námi nově představenou Unit metriku, která operuje nad doménami problémů SAT a TSP. Motivací této naší metriky je namapování výsledku na interval mezi optimální a snadno dostupné řešení (získané hladovým nebo náhodným algoritmem). Namapovaná hodnota bude určovat kvalitu řešení pro každou ze tří složek – *instanci* (U_i), *problém* (U_p) a *experiment* (U_e). K řešení problému různorodosti instancí jsou využity předzpracované informace o nich. V metadatech je pro každou instanci uložena její velikost, název a hodnota optimálního, náhodného a hladového řešení.

První část metriky, instance, využívá informace o optimálním a hladovým řešení. Evaluátor toto řešení namapuje na jednotkový interval [0, 1], kde 0 značí hladové a 1 optimální řešení. Čím blíže je namapovaná hodnota jedničce, tím je řešení bližší optimálnímu a tedy lepší.

$$U_i(instance) = map(instance, optimal, greedy) \in [0, 1] \tag{5.1}$$

Druhá část metriky, problém, využívá znalosti velikosti instancí z metadat. Evaluátor provede vážený průměr nad množinou ohodnocených instancí z první části, váhami jsou zmíněné velikosti instancí.

$$U_p(problem) = weighted\_mean(\{all\, U_i(instance\_i)\}, instances\_sizes) \in [0, 1] \tag{5.2}$$

Třetí část metriky, experiment, nevyužívá metadata. Evaluátor provede průměr ohodnocených domén, které se v experimentu využily, z druhé části metriky. Hodnota experimentu je z intervalu [0, 1] a značí kvalitu algoritmu nad danou množinou instancí.

$$U_e(experiment) = mean(\{all\, U_p(problem)\}) \in [0, 1] \tag{5.3}$$

5.3 Přepočítání leaderboardu

Před samotným konáním mezinárodní výzvy *CHeSC2011* se konalo pravidelné porovnávání přihlášených účastníků. Ti pomocí svých hyper-heuristik řešili řadu optimalizačních problémů implementovaných ve frameworku *HyFlex* a výsledky odesílali ve stanoveném formátu organizátorům. Ti je každý týden seřadili s využitím metriky F1 a tím poskytli přibližný pohled na to, jak bude probíhat samotná výzva.

V následující tabulce (Tabulka 5.1) je zobrazeno poslední vykonané zpracování výsledků účastníků. Tyto data jsme získali z oficiálních stránek výzvy *CHeSC2011*. Pro získání skóre byla využita metrika F1 a nejlépe se umístil algoritmus PHunter4.

Tabulka 5.1 Původní leaderboard

no.	Algoritmus	Autor/Tým	Skóre
1	PHunter4	Fan Xue	204.28
2	ISEA2	Jiri Kubalik	177.28
3	HAHA1	Andreas Lehrbaum	166.78
4	TW4	Hsiao Ping-Che	130.20
5	basic-test	Franco Mascia	125.67
6	ADHS1	Mustafa Misir	120.08
7	SALS	He Jiang	100.33
8	ML	Mathieu Larose	88.67
9	HAEA	Jonatan Gomez	78.07
10	SEA2	David Meignan	69.00
11	AVEG_Nep	Tommaso Urli	63.17
12	MCHH	Kent McClymont	58.50
13	XCJ	Kamran Shafi	46.28
14	FPA1	CS-PUT	43.15
15	SelfSearch	Jawad Elomari	36.00
16	GISS0	Karim A. Touma	34.53
17	Ant-Q	Imen Khamassi	18.00

Díky zveřejněným datům organizátorů výzvy jsme získali přesnou kopii originálních souborů obsahující řešení zmíněných optimalizačních problémů jednotlivých účastníků. Pomocí námi implementovaného evaluátoru v prostředí *HyFlex*, který tento formát dat dokáže zpracovat jsme získali přepočítané skóre pro jednotlivé algoritmy využitím Unit metriky nad doménami problémů TSP a SAT.

Cílem tohoto experimentu je za pomoci námi nově navržené Unit metriky získat objektivní ohodnocení jednotlivých algoritmů a na základě toho změnit jejich pořadí. Data jsme získali z oficiálních stránek výzvy *CHeSC2011* a za pomoci našeho nového evaluátoru využívající Unit metriku zpracovali. Skóre nyní neslouží jen jako hodnota, kterou se pouze určuje pořadí, ale dává informace o tom, jak moc si daný algoritmus vedl. První místo obsadil výherce výzvy Mustafa Misir. Algoritmus PHunter4 se propadl ze svého prvního místa až na sedmé. Ze zajímavých změn lze ještě zmínit ISEA2, které se ze svého druhého místa propadlo pouze o jedno dolů.

V tabulce (Tabulka 5.2) je pro každý algoritmus přepočítáno skóre pomocí Unit metriky.

Tabulka 5.2 Nový leaderboard

no.	Algoritmus	Autor/Tým	Skóre
1	ADHS1	Mustafa Misir	0.8937268288367
2	TW4	Hsiao Ping-Che	0.8717928291134
3	ISEA2	Jiri Kubalik	0.8627590598122
4	basic-test	Franco Mascia	0.8600697728617
5	ML	Mathieu Larose	0.8526865368870
6	HAEA	Jonatan Gomez	0.8505903852983
7	PHunter4	Fan Xue	0.8431685059784
8	SEA2	David Meignan	0.8418202380227
9	XCJ	Kamran Shafi	0.8363892786268
10	HAHA1	Andreas Lehrbaum	0.8356910850024
11	AVEG_Nep	Tommaso Urli	0.7877602004459
12	MCHH	Kent McClymont	0.7778556840891
13	SelfSearch	Jawad Elomari	0.7755255070336
14	SALS	He Jiang	0.7667376760411
15	FPA1	CS-PUT	0.7442084221748
16	GISS0	Karim A. Touma	0.7180372361401
17	Ant-Q	Imen Khamassi	0.6834045643833

5.4 Přepočítání výzvy

Pro vyhodnocení výzvy byla využita sada instancí z trénovací a skryté domény problémů. Z nich se náhodně vybíraly tři trénovací a dvě testovací instance. Nad získanými daty, řešením instancí jednotlivými algoritmy, se provedlo ohodnocení využitím F1 metriky.

V následující tabulce (Tabulka 5.3) je zobrazeno finální pracování výsledků účastníků výzvy *CHeSC2011*. Tyto data jsme získali z oficiálních stránek výzvy. První místo obsadil Mustafa Misir s jeho AdapHH hyper-heuristikou.

Tabulka 5.3 Původní výsledek výzvy

no.	Algoritmus	Autor/Tým	Skóre
1	AdapHH	Mustafa Misir	181.00
2	VNS-TW	Ping-Che Hsiao	134.00
3	ML	Mathieu Larose	131.50
4	PHUNTER	Fan Xue	93.25
5	EPH	David Meignan	89.75
6	HAHA	Andreas Lehrbaum	75.75
7	NAHH	Franco Mascia	75.00
8	ISEA	Jiri Kubalik	71.00
9	KSATS-HH	Kevin Sim	66.50
10	HAEA	Jonatan Gomez	53.50
11	ACO-HH	José Luis Núñez	39.00
12	GenHive	CS-PUT	36.50
13	DynILS	Mark Johnston	27.00
14	SA-ILS	He Jiang	24.25
15	XCJ	Kamran Shafi	22.50
16	AVEG-Nep	Tommaso Urli	21.00
17	GISS	Alberto Acuña	16.75
18	SelfSearch	Jawad Elomari	7.00
29	MCHH-S	Kent McClymont	4.75
20	Ant-Q	Imen Khamassi	0.00

Webové stránky organizátora výzvy sice neposkytují surové soubory s uloženým řešením instancí jednotlivými algoritmy, ale zato poskytují tabulku, ve které jsou pro každý algoritmus uložena řešení jednotlivých instancí. Dále i jejich průměrná hodnota, ale ta je pro tuto práci irelevantní. Pomocí jednoduchého crawleru jsme tyto data získali a zapsali do souboru s pevnou strukturou, které rozumí framework *HyFlex*.

Cílem tohoto experimentu je za pomoci námi nově navržené Unit metriky získat objektivní ohodnocení jednotlivých algoritmů a na základě toho změnit jejich pořadí. Data jsme získali crawlerem z oficiálních stránek výzvy a poté jsme provedli přepočet pomocí nové Unit metriky pro domény problémů TSP a SAT a výsledky zapsali do následující tabulky (Tabulka 5.4).

Využitím nového skóre se výherce výzvy, algoritmus AdapHH, propadl na druhé místo a prvenství vystřídal algoritmus VNS-TW, který býval na místě druhém. Podobná výměna nastala taktéž na místě třetím a čtvrtém. Algoritmus, který si polepšil o více než jedno místo, je například ISEA, jež se z osmého místa dostal na páté.

Tabulka 5.4 Nový výsledek výzvy

no.	Algoritmus	Autor/Tým	Skóre
1	VNS-TW	Ping-Che Hsiao	0.9209100846675127
2	AdapHH	Mustafa Misir	0.9134175556568159
3	PHUNTER	Fan Xue	0.9056115249942573
4	ML	Mathieu Larose	0.9052700908058408
5	ISEA	Jiri Kubalik	0.9033113381662342
6	HAEA	Jonatan Gomez	0.8991415824946434
7	NAHH	Franco Mascia	0.8977335479645917
8	EPH	David Meignan	0.8900785702586111
9	HAHA	Andreas Lehrbaum	0.8805259602196633
10	XCJ	Kamran Shafi	0.8731287407864694
11	ACO-HH	José Luis Núñez	0.8694754933675708
12	AVEG-Nep	Tommaso Urli	0.8681296983724813
13	KSATS-HH	Kevin Sim	0.8638867680781335
14	SelfSearch	Jawad Elomari	0.8604501026957524
15	GenHive	CS-PUT	0.8578604007342212
16	MCHH-S	Kent McClymont	0.8427543952271577
17	DynILS	Mark Johnston	0.8331571840028011
18	SA-ILS	He Jiang	0.8305936334655835
19	GISS	Alberto Acuña	0.8165953689408881
20	Ant-Q	Imen Khamassi	0.7884620960447779

5.5 První experiment [30 sekund]

V tomto experimentu jsme postupně spustili algoritmy nad pěticí instancí domén problémů TSP a SAT, které se využily i v mezinárodní výzvě a z těchto běhů vybrali nejlepší řešení.

Cílem tohoto experimentu je s exponenciálně se zvyšujícím opakováním časově omezeného výpočtu jednotlivých instancí optimalizačních problémů na dobu 30 sekund sledovat s využitím Unit metriky změny kvality řešení algoritmů. V první části se nachází pětice hyper-heuristik s kterými zbylé algoritmy porovnáváme. V poslední řadě porovnáváme i nově implementovanou hyper-heuristiku.

Z tabulky (Tabulka 5.5) lze vyčíst, že takto omezený čas se zvyšujícím se opakováním nedává lepší řešení. Nejlépe z tohoto experimentu dopadla GIHH (AdapHH), která se skórem 0.838 překonala ostatní. Na druhém místě skončila LeanGIHH. Z metaheuristik skončila nejlépe TS (Tabu Search), jež i přesto, že není hyper-heuristikou získala více než polovinu skóre ISEA ve vybraných opakováních. Nová hyper-heuristika se i přes svou jednoduchou konstrukci držela těsně pod polovinou skóre ISEA.

Tabulka 5.5 Testování algoritmů na datech výzvy CHeSC2011 s časem výpočtu 30 s

	Opakování			
Algoritmus	**4**	**8**	**16**	**32**
PearlHunter	0.7758103442550	0.7810349046510	0.7830950849930	0.7764164030490
EPH	0.8097632618730	0.7857823483700	0.7687958339300	0.7604737319870
GIHH	0.8500510667580	0.8321993829540	0.8380195822260	0.8380778641630
LeanGIHH	0.8305090714890	0.8119919050520	0.8343833854250	0.8232339026390
ISEA	0.7753020520800	0.7950590013740	0.7791388681280	0.7672924846020
GA	0.1125478950403	0.0373600985796	0.0630032339968	0.0732043306114
TS	0.3371518056379	0.3572579265830	0.3708024733582	0.3635665868275
SA	0.0624315474200	0.0687227745862	0.0748117498041	0.1055498964760
ACO	0.0121993292380	0.0621678102013	0.0191089862321	0.0121993292380
new-hh	0.3845949224473	0.3608524568073	0.3226217908442	0.3428384380278

5.6 Druhý experiment [75 sekund]

Cílem tohoto experimentu je s exponenciálně se zvyšujícím opakováním výpočtu jednotlivých instancí optimalizačních problémů na dobu 75 sekund sledovat s využitím Unit metriky změny kvality řešení algoritmů na doménách problémů SAT a TSP.

V tabulce (Tabulka 5.6) lze pozorovat, že se zvýšeným časem se zlepšilo i skóre většiny algoritmů. Na prvním místě se opět umístil GIHH (AdapHH) a těsně za ním LeanGIHH. U metaheuristik došlo k výraznému zlepšení u TS (Tabu Search), který opět většinou získal lehce pod polovinu skóre PearlHunter. Nová hyper-heuristika si oproti minulému běhu nijak zásadně nepolepšila, ale zůstala nad hranicí 0.31.

Tabulka 5.6 Testování algoritmů na datech výzvy CHeSC2011 čas 75 s

	Opakování			
Algoritmus	**4**	**8**	**16**	**32**
PearlHunter	0.8095055127130	0.8000242774580	0.8025513451750	0.7938015942350
EPH	0.7913412616240	0.8160425557580	0.7955154690680	0.8071429423910
GIHH	0.8683874148740	0.8678861728620	0.8517135778700	0.8544528990000
LeanGIHH	0.8260520828540	0.8555989874830	0.8354123229150	0.8338692777450
ISEA	0.7926971717980	0.8200417875250	0.7982418471400	0.8025321206510
GA	0.1476198324174	0.0286994858790	0.1480449942318	0.0555525945010
TS	0.3962879434410	0.3559954336400	0.3671044127800	0.3821834763490
SA	0.0995699366241	0.0935792863938	0.1267459450106	0.0905105633418
ACO	0.0071733345083	0.0076549271000	0.0650095431334	0.0300500121356
new-hh	0.3473288273593	0.3262922996429	0.3462478948685	0.3160945014795

5.7 Třetí experiment [150 sekund]

Cílem tohoto experimentu je s exponenciálně se zvyšujícím opakováním výpočtu jednotlivých instancí optimalizačních problémů na dobu 150 sekund sledovat s využitím Unit metriky změny kvality řešení algoritmů na doménách problémů SAT a TSP.

Z tabulky (Tabulka 5.7) lze vyčíst zlepšení jednotlivých hyper-heuristik, kde znovu exceloval GIHH (AdapHH) s LeanGIHH, který jej překonalo při 16 opakováních. Nejlepší metaheuristikou opět skončila TS (Tabu Search) s nejlepším skórem 0.40. Nová hyper-heuristika opět skončila se skoro polovinou skóre PearlHunter.

Tabulka 5.7 Testování algoritmů na datech výzvy CHeSC2011 s časem výpočtu 150 s

	Opakování			
Algoritmus	**4**	**8**	**16**	**32**
PearlHunter	0.8157826075260	0.8204142127990	0.8138423191000	0.8179471213550
EPH	0.8327225053800	0.8278831413560	0.8125503196960	0.8246861108290
GIHH	0.8925726656940	0.8854706223460	0.8783690786780	0.8829693541500
LeanGIHH	0.8867663477330	0.8669773575200	0.8826703841710	0.8681613090310
ISEA	0.8483242905400	0.8323121164280	0.8324804341030	0.8315205508200
GA	0.0826304309846	0.0688611758070	0.1632086424126	0.1792631126510
TS	0.3526937369230	0.3707728374370	0.3571066694857	0.4006084888280
SA	0.1199801367218	0.1176232657090	0.0726236556182	0.1707099041149
ACO	0.0060561569890	0.0037597776700	0.0082581745420	0.0083940371380
new-hh	0.3467150824319	0.3718735180811	0.3519913049587	0.3573602282952

5.8 Čtvrtý experiment [300 sekund]

Posledním experiment jsme spouštěli s časem, který byl podobný tomu použitém při mezinárodní výzvě s omezením se na domény problémů TSP a SAT. Jedná se tedy o simulaci výzvy samotné na našem systému.

Cílem tohoto experimentu je s exponenciálně se zvyšujícím opakováním výpočtu jednotlivých instancí optimalizačních problémů na dobu 300 sekund sledovat s využitím Unit metriky změny kvality řešení algoritmů.

Z výsledků tabulky (Tabulka 5.8) lze opět vyčíst, že nejlepší hyper-heuristikou byl GIHH (AdapHH), jež ve většině testů oproti ostatním překonal hranici skóre 0.90. Těsně pod ním se znovu umístil LeanGIHH se ztrátou 0.01 u většiny opakování.

Z metaheuristik zas nad ostatními exceloval TS (Tabu Search), který následoval s polovičním skórem GA (Genetic Algorithm). Nová hyper-heuristika opět stála těsně pod polovinou skóre PearlHunter.

Tabulka 5.8 Testování algoritmů na datech výzvy CHeSC2011 s časem výpočtu 300 s

	Opakování			
Algoritmus	**4**	**8**	**16**	**32**
PearlHunter	0.8327491369110	0.8397504711290	0.8284752721080	0.8326800446400
EPH	0.8219319589540	0.8439415608630	0.8453678530240	0.8501219514630
GIHH	0.9057947852880	0.9022764923980	0.9050461682050	0.8988274227320
LeanGIHH	0.8992858179940	0.8974113285900	0.8695498723960	0.8893368805790
ISEA	0.8586290169910	0.8622835560260	0.8508447804980	0.8513198497170
GA	0.1211317813146	0.2260168313753	0.2468940081040	0.2375476106203
TS	0.3741084075119	0.3828696250690	0.4060168039560	0.4209877473489
SA	0.0444053133998	0.0771687319899	0.1090404208000	0.0753308624955
ACO	0.0070907134207	0.0084720982259	0.0142414387478	0.0478242566423
new-hh	0.3637851614567	0.3747708923343	0.3647980695488	0.3704925417740

5.9 Experiment nad doménami problémů

Samotné srovnání pomocí všech instancí postrádá nahlédnutí do toho, jak si jednotlivé experimenty vedly. Proto jsme v tomhle experimentu zachytili skóre jednotlivých algoritmů nad jednotlivými instancemi.

Cílem tohoto experimentu je sledovat s využitím Unit metriky kvalitu algoritmů na jednotlivých instancích domény problému TSP.

Použitá data jsou z předchozího experimentu nejvyššího počtu opakování, tedy s časem 300 sekund a 32 opakováním výpočtů nad jednou instancí.

Z následující tabulky (Tabulka 5.9) lze vyčíst, že oproti skóre přes všechny domény problémů zde GIHH (AdapHH) občas ztratilo své prvenství. U instancí rat575 a d1291 jej překonal EPH, u instance usa13509 pak LeanGIHH.

Z výsledků metaheuristik a nové hyper-heuristiky je patrné, že na doméně TSP nejsou jejich výsledky zrovna moc dobré. Nulovové skóre značí, že algoritmus na dané instanci nedosahuje lepších výsledků než hladový algoritmus.

Tabulka 5.9 Testování algoritmů na instancích TSP výzvy CHeSC2011 s časem výpočtu 300 s

	Instance				
Algoritmus	**rat575**	**d1291**	**pr299**	**usa13509**	**u2152**
PearlHunter	0.97138313	0.75609414	0.99976906	0.75344484	0.77596923
EPH	0.97845664	0.81587777	0.99976906	0.79276062	0.83892396
GIHH	0.97804670	0.80433640	0.99976906	0.84336617	0.83338370
LeanGIHH	0.97516227	0.75769744	0.99976906	0.84781750	0.83153891
ISEA	0.95965110	0.77570778	0.99976906	0.82099709	0.80647770
GA	0.18286099	0.19907718	0.32206185	0.07852843	0.11047860
TS	0.15047233	0.19753917	0.40217967	0.07946793	0.22925012
SA	0.32321187	0.23598961	0.39711340	0.07049424	0.13338955
ACO	0.00000000	0.28222628	0.48594992	0.00000000	0.15932251
new-hh	0.06680161	0.00000000	0.15870397	0.01994413	0.01876449

Cílem tohoto experimentu je sledovat s využitím Unit metriky kvalitu algoritmů na jednotlivých instancích domény problému SAT.

Pozorováním následující tabulky (Tabulka 5.10) lze vyčíst, že z hyper-heuristik si první místo drží GIHH (AdapHH) a těsně za ním LeanGIHH. Zajímavým pozorováním u metaheuristik je, že TS (Tabu Search) sama o sobě držela těsný krok s ostatními hyper-heuristikami a u instance jarv-684-2300 většinu z nich překonala. Nová hyper-heuristika se u většiny instancí držela těsně za ISEA.

Nulovové skóre značí, že algoritmus na dané instanci nedosahuje lepších výsledků než hladový algoritmus.

Tabulka 5.10 Testování algoritmů na instancích SAT výzvy CHeSC2011 s časem výpočtu 300 s

	Instance				
Algoritmus	**pg-525 -2336**	**pg-696 -3122**	**hg4-300 -1200**	**pg-525 -2276**	**jarv-684 -2300**
PearlHunter	0.94186046	0.89565217	0.76595744	0.92307692	0.90476190
EPH	0.94186046	0.88695652	0.76595744	0.91025641	0.89523809
GIHH	0.97674418	0.94782608	0.82978723	0.96153846	0.98095238
LeanGIHH	0.97674418	0.92173913	0.82978723	0.94871794	0.94285714
ISEA	0.93023255	0.86956521	0.78723404	0.89743589	0.87619047
GA	0.52325580	0.46956520	0.57446800	0.00000000	0.37142857
TS	0.53488372	0.53043478	0.78723404	0.82051282	0.98095238
SA	0.24418604	0.00000000	0.00000000	0.00000000	0.00000000
ACO	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
new-hh	0.75581395	0.51304347	0.65957446	0.82051282	0.84761904

5.10 Shrnutí experimentů

Ze získaných dat experimentů je patrné, že samotné metaheuristiky, které ve frameworku *SEAGE* slouží jakožto stavební bloky pro tvorbu hyper-heuristik, nedosahují podobných výsledků jako hyper-heuristiky z výzvy *CHeSC2011*. Když ale srovnáme výsledky jejich jednotlivých instancí optimalizačních problémů, tak pro doménu problému SAT jsou s nimi občas srovnatelné, především tedy Tabu Search (TS). To platí i pro nově představenou hyper-heuristiku, jejíž implementace je pouze na několika málo řádek kódu. Framework *SEAGE* má tedy veliký potenciál v oblasti hyper-heuristik, díky využití již samostatně velmi výkoných metaheuristik.

Kapitola 6
Závěr

Hlavním cílem této práce bylo zhodnocení optimalizačního frameworku *SEAGE* z pohledu aktuálního stavu výzkumu v oblasti hyper-heuristik. K tomu jsme v práci zrekonstruovali framework *HyFlex*, který byl využit v mezinárodní výzvě *CHeSC2011* proto, abychom mohli využít hyper-heuristiky účastníků této výzvy k srovnání se stavebními bloky hyper-heuristik (metaheuristiky) z frameworku *SEAGE* tak, aby veškeré výpočty probíhaly na jednom systému a výsledky byly tedy co nejvěrohodnější. I přes úspěšné vytvoření rekonstrukce frameworku *HyFlex* jsme museli použitý evaluátor i s metrikou opustit. Nevyhovoval našim požadavkům na objektivní zhodnocení řešení jednotlivých algoritmů nad jednotlivými instancemi domén optimalizačních problémů.

Pro tyto důvody jsme tedy představili a implementovali Unit metriku společně s evaluátorem, který ji využívá. Díky jejímu návrhu ji nebyl problém zahrnout do frameworku *SEAGE* ani do upraveného prostředí frameworku *HyFlex*. Nad frameworkem *HyFlex* jsme vytvořili prostředí, které jej využívá pro spouštění hyper-heuristik a dále umožňuje spuštění nového i starého evaluátoru nad vybranou množinou souborů s řešením.

Ve frameworku *SEAGE* jsme díky otevřenosti prostření rozšířili Unit metriku ještě o parametr zlepšení. Oba tyto parametry pak poskytují nové informace pro rozhodovací proces hyper-heuristik. Pro představení tohoto procesu i implementace hyper-heuristik v prostředí *SEAGE* jsme jednu základní vytvořili tak, aby mohla sloužit jako základ pro další hyper-heuristiky.

S využitím ohodnocení pomocí nové Unit metriky jsme provedli sérii experimentů, ve kterých ukazujeme, že i přes poměrně horší celkové skóre v porovnání s konkurenčními hyper-heuristikami si už samotné metaheuristiky nevedou vůbec špatně. Pokud porovnání pak přeneseme na jednotlivé instance domény problému SAT, tak můžeme pozorovat velice dobré výsledky již základních metaheuristik i samotné nové jednoduché hyper-heuristiky. Framework *SEAGE* má tedy veliký potenciál v oblasti hyper-heuristik, díky využití již samostatně velmi výkonných metaheuristik.

Pro budoucí práci by bylo zajímavé experimentovat s vytvořením sofistikovanějších hyper-heuristik inspirovaných například těmi z výzvy *CHeSC2011* pro řešení optimalizačních problémů. Nejdříve je ale potřeba ve framework *SEAGE* rozšířit podporu všech implementovaných algoritmů na ostatní domény problémů.

Bibliografie

1. SÖRENSEN, Kenneth; SEVAUX, Marc; GLOVER, Fred. A History of Metaheuristics. *Handbook of Heuristics* [online]. 2017, s. 1–6 [cit. 2021-04-18]. Dostupné z: `https://www.researchgate.net/publication/315811561_A_History_of_Metaheuristics`.
2. SÖRENSEN, Kenneth; SEVAUX, Marc; GLOVER, Fred. A History of Metaheuristics. *Handbook of Heuristics* [online]. 2017, s. 19–20 [cit. 2021-04-20]. Dostupné z: `https://www.researchgate.net/publication/315811561_A_History_of_Metaheuristics`.
3. DRAKE, John H.; KHEIRI, Ahmed; ÖZCAN, Ender; BURKE, Edmund K. Recent advances in selection hyper-heuristics. *European Journal of Operational Research* [online]. 2020, roč. 285, č. 2, s. 405–428 [cit. 2021-04-19]. ISSN 0377-2217. Dostupné z DOI: `https://doi.org/10.1016/j.ejor.2019.07.073`.
4. NOTTINGHAM, University of. CHeSC 2011 Cross-domain Heuristic Search Challenge. In: *Documentation* [online]. 2011 [cit. 2021-04-21]. Dostupné z: `http://www.asap.cs.nott.ac.uk/external/chesc2011/index.html`.
5. OCHOA, G.; HYDE, M.; CURTOIS, T.; VAZQUEZ-RODRIGUEZ, J.A.; WALKER, J.; GENDREAU, M.; KENDALL, G.; MCCOLLUM, B.; PARKES, A.J.; PETROVIC, S.; BURKE, E.K. HyFlex: A Benchmark Framework for Cross-domain Heuristic Search. In: HAO, J.-K.; MIDDENDORF, M. (ed.). *European Conference on Evolutionary Computation in Combinatorial Optimisation(EvoCOP 2012)* [online]. Heidelberg: Springer, 2012, sv. 7245, s. 136–147 [cit. 2021-05-03]. LNCS. Dostupné z: `http://www.asap.cs.nott.ac.uk/external/chesc2011/reports/HyFlexTutorialEvoCOP2012.pdf`.
6. ADRIAENSEN, Steven; NOW'E, Ann. Case Study: An Analysis of Accidental Complexity in a State-of-the-art Hyper-heuristic for HyFlex. In: *Evolutionary Computation (CEC), 2016 IEEE Congress on* [online]. 2016 [cit. 2021-04-20]. Dostupné z: `https://github.com/Steven-Adriaensen/Lean-GIHH`.
7. MISIR, M.; CAUSMAECKER, P. De; BERGHE, G. Vanden; VERBEECK, K. Google Code Archive. In: *An adaptive hyper-heuristic for CHeSC 2011* [online]. 2011 [cit. 2021-04-21]. Dostupné z: `https://code.google.com/archive/p/generic-intelligent-hyper-heuristic`.
8. BURKE, Edmund K; GENDREAU, Michel; HYDE, Matthew; KENDALL, Graham; OCHOA, Gabriela; ÖZCAN, Ender; QU, Rong. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society* [online]. 2013, roč. 64, č. 12, s. 1695–1724 [cit. 2021-04-21]. Dostupné z DOI: `10.1057/jors.2013.71`.

9. BRANKE, Jürgen; NGUYEN, Su; PICKARDT, Christoph W.; ZHANG, Mengjie. Automated Design of Production Scheduling Heuristics: A Review. *IEEE Transactions on Evolutionary Computation* [online]. 2016, roč. 20, č. 1, s. 110–124. ISSN 1941-0026. Dostupné z DOI: 10.1109/TEVC.2015.2429314.
10. BOYD, Stephen; VANDENBERGHE, Lieven. *Convex Optimization* [online]. Cambridge University Press, 2004. ISBN 978-0-521-83378-3. Dostupné také z: https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf.
11. SUNG, Phil. Maximum Satisfiability. In: *Maximum Satisfiability* [online]. 2006, s. 1–2 [cit. 2021-04-30]. Dostupné z: http://math.mit.edu/~goemans/18434S06/max-sat-phil.pdf.
12. MA, Suzanne. *Understanding The Travelling Salesman Problem (TSP)* [online]. 2020 [cit. 2021-04-30]. Dostupné z: https://blog.routific.com/travelling-salesman-problem.
13. COOK, Stephen A. An overview of computational complexity. *Communications of the ACM* [online]. 1983, roč. 26, s. 401–408 [cit. 2021-05-03]. Dostupné z DOI: 10.1145/1283920.1283938.
14. BERGSTRA, James; BENGIO, Yoshua. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* [Online]. 2012, roč. 13, č. 1, s. 281–305 [cit. 2021-05-03]. ISSN 1532-4435. Dostupné z: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.
15. MOORE, Karleigh; KHIM, Jimin; ROSS, Eli. *Greedy Algorithms* [online]. 2020 [cit. 2021-05-01]. Dostupné z: https://brilliant.org/wiki/greedy-algorithm.
16. BLUM, Christian; ROLI, Andrea. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput. Surv.* [Online]. 2001, roč. 35, s. 268–308. Dostupné z DOI: 10.1145/937503.937505.
17. ROETZEL, Wilfried; LUO, Xing; CHEN, Dezhen. Chapter 6 - Optimal design of heat exchanger networks. In: ROETZEL, Wilfried; LUO, Xing; CHEN, Dezhen (ed.). *Design and Operation of Heat Exchangers and their Networks* [online]. Academic Press, 2020, s. 231–317 [cit. 2021-05-01]. ISBN 978-0-12-817894-2. Dostupné z DOI: https://doi.org/10.1016/B978-0-12-817894-2.00006-6.
18. GLOVER, Fred. Tabu Search – Part 1. *ORSA Journal on Computing.* [Online]. 1989, roč. 1, s. 190–206. Dostupné z DOI: 10.1287/ijoc.1.3.190.
19. LEE, Jacobson. *Simulated Annealing* [online]. 2013 [cit. 2021-05-02]. Dostupné z: https://www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/6.
20. KUBALÍK, Jiří. *Optimalizační algoritmy inspirované chováním mravenců* [online]. 2004 [cit. 2021-05-02]. Dostupné z: http://labe.felk.cvut.cz/~kubalik/kopr/kopr_ants.pdf.
21. BURKE, Edmund; KENDALL, Graham; NEWALL, Jim; HART, Emma; ROSS, Peter; SCHULENBURG, Sonia. Hyper-Heuristics: An Emerging Direction in Modern Search Technology. In: [online]. 2003, s. 457–474. Dostupné z DOI: 10.1007/0-306-48056-5_16.
22. BURKE, Edmund; ERBEN, Wilhelm. *A Hyperheuristic Approach to Scheduling a Sales Summit* [online]. 2001. ISBN 978-3-540-44629-3. Dostupné z DOI: 10.1007/3-540-44629-X.
23. MEIGNAN, David. *An Evolutionary Programming Hyper-heuristic with Co-evolution for CHeSC'11* [online]. 2011 [cit. 2021-05-03]. Dostupné z: http://www.asap.cs.nott.ac.uk/external/chesc2011/entries/meignan-chesc.pdf.
24. MISIR, M.; DE CAUSMAECKER, P.; VANDEN BERGHE, G.; VERBEECK, K. *An adaptive hyper-heuristic for CHeSC 2011* [online]. 2011 [cit. 2021-05-03]. Dostupné z: http://www.asap.cs.nott.ac.uk/external/chesc2011/entries/misir-chesc.pdf.

25. KUBALÍK, Jiří. *Iterated Search Driven by Evolutionary Algorithm Hyper-Heuristic* [online]. 2011 [cit. 2021-05-03]. Dostupné z: http://www.asap.cs.nott.ac.uk/external/chesc2011/entries/kubalic-chesc.pdf.

26. MAFAN, Xue; CHAN, C.Y.; IP, W.H; CHEUNG, C.F. *Pearl Hunter: A Hyper-heuristic that Compiles Iterated Local Search Algorithms* [online]. 2011 [cit. 2021-05-03]. Dostupné z: http://www.asap.cs.nott.ac.uk/external/chesc2011/entries/xue-chesc.pdf.

27. MÁLEK, Richard. *Basics* [online]. 2021 [cit. 2021-05-03]. Dostupné z: https://www.seage.org/#basics.

28. XANDRI, Juan Pablo. NOTES ON METRIC SPACES. In: [online]. 2015, s. 1–2 [cit. 2021-05-04]. Dostupné z: https://scholar.princeton.edu/sites/default/files/jxandri/files/notes_on_metric_spaces_0.pdf.

29. OMRAI, David; MÁLEK, Richard. *Extended reproduction of the HyFlex framework with a few hyper-heuristics from competition CHeSC2011* [online]. 2021 [cit. 2021-05-11]. Dostupné z: https://github.com/seage/hyflex.

30. MÁLEK, Richard; OMRAI, David; DURKOTA, Karel; ZMÁTLÍK, Jan. *SEAGE* [online]. 2010 [cit. 2021-05-11]. Dostupné z: https://github.com/seage/seage.

Obsah přiloženého média