Combining logic and optimization for applications in probabilistic and non-monotonic reasoning

Tatjana Stojanović

University of Kragujevac, Faculty of Science
Radoja Domanovića 12, 34000 Kragujevac, Serbia
Emails: tanjat@kg.ac.rs, tatjana.stojanovic@pmf.kg.ac.rs

Abstract

This paper presents an overview of heuristic methods for solving satisfiability problems in two probabilistic logics LPP_2^{ext} and $LPCP$ and System **P**. These systems are very suitable for representing and reasoning with uncertain knowledge and for modeling default reasoning. Considered representations enable reducing the satisfiability problem to a problem of finding solution for system of lienar inequalities. The complexity of the problems considered does not allow efficient application of exact methods for fining solutions, but imposes a heuristic approach. Experimental evaluation shows a high success rate in proving the satisfiability of randomly generated formulas.
Key words: Uncertain knowledge, Probabilistic logic, Non-monotonic reasoning, Heuristics

1 Introduction

Representing uncertain knowledge and reasoning over that knowledge is a problem present in mathematical logic and computer science since the first works of Leibnitz and Bool. Many of the formalisms for representing and reasoning with uncertainty are based on probabilistic logics [1, 2, 3, 4]. Since 1980 various systems have been developed for non-monotonic reasoning [5, 6]. In those systems conclusions are drawn defeasibly and the right to retract them in the light of further information is reserved. In order to develop algorithms for automated reasoning, it is necessary for the observed logical system to be sound and complete. In many logics, drawing a set of conclusions can be reduced to a satisfiability problem. In most of logic systems with mentioned properties, satisfiabilty problem is NP-complete. Some restricted subsets can be P-complete.

Using exact algorithms for solving satisfiability problems for real-life situations represented by some logic system is inadequate and can't produce an answer in reasonable time. That is the main reason for using heuristics approach in dealing with this type of problems. Using meta-heuristic for solving SAT or MAXSAT problem in propositional logic is well studied and documented. Various types of heuristics based on local search, genetic algorithm and swarm intelligence are applied to those problems. Significantly fewer papers are devoted to the use of heuristic methods for solving satisfiability problems in systems based on probability logics and defeasible systems. The main part of this paper will be devoted to the presentation of some such systems and the results obtained by using heuristic methods to solve the satisfiability problem in these systems. Most of presented results are published in [7, 8].

The rest of the paper is organized as follows. Section 2 contains basic notation in logic systems supporting probability and non-monotonic reasoning. In Section 3 experimental results of applying meta-heuristics on satisfiability problems in given systems

are presented. Section 4 contains conclusions and directions for future work.

2 Mathematical background

In this section we will present System **P** consisting of rules aimed to capture reasoning with defeasible assertions (defaults) and some probabilistic logics obtained extending propositional logic with two different probabilistic operators.

Let For_C be the set of classical propositional formulas constructed from a set of propositional letters Var (denoted by the lower case letters p, q, r, ...) and the usual connectives: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$. Classical formulas are denoted by the Greek letters: α, β, γ ... A world v is a truth assignment to the propositional letters (a mapping $v : \text{Var} \rightarrow \{0, 1\}$) which is inductively extended to all classical formulas as usual. If W is a set of worlds, we denote by $[\alpha]_W$ the set of all worlds in which α is true, i.e., $[\alpha]_W = \{v \in W : v(\alpha) = 1\}$. Let A be a probabilistic formula defined below, specific for each logic, and $p_1, \ldots, p_n$ be the list of all propositional letters from A. An atom a of A is a formula $\pm p_1 \wedge \ldots \wedge \pm p_n$. Note that all pairs of different atoms are mutually exclusive. We use $At(A)$ to denote the set of all atoms from A, and n to denote the number of propositional letters from A. Obviously, $|At(A)| = 2^n$.

For every system described below it is necessary to define the probability model as a tuple of the form $\mathbf{M} = \langle W, H, \mu \rangle$, where W is a set of worlds, H is a field (algebra) of subsets of W containing $[\alpha]_W$, for every formula $\alpha \in \text{For}_C$, and μ is a finitely additive probability measure, such that only the empty set has the zero probability.

2.1 Logic LPP_2^{ext}

Logic LPP_2^{ext} is introduced in papers [1, 9]. Probability operator P^* is added to a previously defined set of symbols.

The set of formulas consists of:

- the set of all classical propositional formulas denoted by For_C;
- the set of weighted formulas denoted by For_W (the formulas from the set For_W will be denoted by A, B, C, ...) which are built as follows:
  - a primitive weighted term is an expression in the form $P^*(\alpha)$, where α is a propositional formula;
  - a weighted term is an expression in the form $a_1P^*(\alpha_1) + a_2P^*(\alpha_2) + \cdots + a_kP^*(\alpha_k)$, where $a_1, a_2, \ldots a_k \in Q$ and $k \geq 1$;
  - a basic weighted formulas are expressions in the form of $t \geq s$, where t is a weighted term and $s \in Q$;
  - if A and B are weighted formulas then $A \wedge B$ and $\neg A$ are weighted formulas.

The logic defined in the given way can be used to reason about probabilities. For example, the sentence "probability of (event) p is less than $\frac{1}{3}$ and p is at least twice as probable as q" can be expressed with $(3P^*(p) < 1) \wedge (P^*(p) - 2P^*(q) > 0)$. The formula $P^*(p) > \frac{1}{2}P^*(q)$ can be used to describe the assertion "the conditional probability of p given q is at least $\frac{1}{2}$".

Avoiding all details given in paper [9] we can say that a formula

$$a_1P^*(\alpha_1) + a_2P^*(\alpha_2) + \cdots + a_kP^*(\alpha_k) \geq s$$

is satisfiable in model M iff

$$\sum_{i=1}^{k} a_i \mu([\alpha_i]) \geq s.$$

Using previous statement it is possible to reduce satisfiability problem to a problem of solving linear system of inequalities. Existence of solution of corresponding system of inequalities guarantees satisfiabilty of the considered set of formulas. Satisfiability problem in this logic is usually denoted by PSAT.

2.2 Logic LPCP

The Hardy field $Q(\varepsilon)$ is a recursive non-archimedean field which contains all rational functions of a fixed positive infinitesimal ε which belongs to a nonstandard elementary extension ${}^*R$ of the standard real numbers ([10, 11]). An element ε of ${}^*R$ is an infinitesimal if $|\varepsilon| < \frac{1}{n}$ for every natural number n. Some examples of infinitesimal are (in ascending order, if $\varepsilon > 0$): $\varepsilon^3 + \varepsilon^4$, $\varepsilon^2 - 5\varepsilon^6$, $\frac{\varepsilon}{100}$, 85ε, or negative infinitesimals: $-\varepsilon$, $-\varepsilon^2$, ...

$Q(\varepsilon)$ contains all rational numbers. Let S be the unit interval of the $Q(\varepsilon)$ and $Q[0,1]$ denote the set of rational numbers from $[0,1]$. Each $\eta \in \mathbb{Q}(\varepsilon)$ can be expressed in the normalized form:

$$(*) \qquad \eta = \frac{a\varepsilon^k + \sum_{i=k+1}^{n} a_i \varepsilon^i}{1 + \sum_{j=1}^{m} b_j \varepsilon^j}, \quad k < n,\ 0 < m,$$

for some unique integer k and some unique leading coefficient a such that $a \neq 0$ unless $\eta = 0$. The ordering $<$ on $\mathbb{Q}(\varepsilon)$ is defined by: $\eta > 0$ iff $a > 0$.

Let $(CP_{\leqslant s})_{s\in S}$, $(CP_{\geqslant s})_{s\in S}$, and $(CP_{\approx r})_{r\in Q[0,1]}$ be the binary probabilistic operators. The set For_P^S of probabilistic propositional formulas is the smallest set Y containing all formulas of the forms:

- $CP_{\geqslant s}(\alpha, \beta)$ for $\alpha, \beta \in For_C$, $s \in S$,
- $CP_{\leqslant s}(\alpha, \beta)$ for $\alpha, \beta \in For_C$, $s \in S$ and
- $CP_{\approx r}(\alpha, \beta)$ for $\alpha, \beta \in For_C$, $r \in Q[0,1]$,

and closed under the formation rules: if A belongs to Y, then $\neg A$ is in Y, and if A and B belong to Y, then $(A \wedge B)$ is in Y. Note that neither mixing of pure propositional formulas and probabilistic formulas, nor nested probabilistic operators are allowed. For example, $\alpha \wedge CP_{\geqslant s}(\alpha, \beta)$ and $CP_{\leqslant s}(\alpha, CP_{\geqslant r}(\beta, \gamma))$ are not well formed formulas, while $CP_{>0.5+\varepsilon}(p \wedge q \wedge r, p \vee r) \wedge CP_{\leqslant 0.8-\varepsilon^2}(\neg p \wedge r, \neg p)$ and $\neg CP_{\approx 0.75}((p \vee q) \to r, \neg p \wedge \neg q)$ are examples of correct formulas.

Let $\mu : At(A) \to S$ be a probability measure. We introduce the following abbreviations:

- x_i denotes the measure of the atom $a_i \in At(A), i = 1, \ldots, 2^n$;
- $\sum(\alpha)$ denotes $\sum_{a_i \in At(A): a_i \models \alpha} x_i$, and
- $\mathrm{C}\sum(\alpha, \beta)$ denotes $\frac{\sum(\alpha \wedge \beta)}{\sum(\beta)}$.

formula A is satisfiable if the following holds:

1. if $A \in For_C$ it is satisfiable if $(\exists a_i \in At(A))a_i \models A$,
2. if A has a form $CP_{\leqslant s}(\alpha, \beta)$ it is satisfiable if either $\sum(\beta) = 0$ and $s = 1$ or $\sum(\beta) > 0$ and $\mathrm{C}\sum(\alpha, \beta) \leqslant s$,
3. if A has a form $CP_{\geqslant s}(\alpha, \beta)$ it is satisfiable if either $\sum(\beta) = 0$ or $\sum(\beta) > 0$ and $\mathrm{C}\sum(\alpha, \beta) \geqslant s$,
4. if A has a form $CP_{\approx r}(\alpha, \beta)$ it is satisfiable if either $\sum(\beta) = 0$ and $r = 1$ or $\sum(\beta) > 0$ and for every positive integer n, $\mathrm{C}\sum(\alpha, \beta) \in [\max(0, r - \frac{1}{n}), \min(1, r + \frac{1}{n})]$,
5. $\neg A$ is satisfiable if A is not satisfiable,
6. $A \wedge B$ is satisfiable if A is satisfiable and B is satisfiable.

A detailed explanation of how satisfiability problem in $LPCP$ can be transformed into a problem of finding a solution for a system of linear inequalities is given in [4, 7]. Satisfiability problem in this logic is usually denoted by CPSAT-ε.

2.3 System **P**

Let $\rightarrowtail$ be a new binary logical connective. If α, $\beta \in \mathrm{For}_C$, a default (or: conditional assertion) is the formula $\alpha \rightarrowtail \beta$, with the intended meaning:

- β normally (or: typically, generally, usually) follows from α,
- β is a plausible (or: defeasible) consequence of α, etc.

The set $\mathrm{For}_{\rightarrowtail}$ contains all Boolean combinations of defaults.

The most famous example of this type of reasoning is:

- if *birds fly*, *penguins are birds* and *penguins don't fly*, could we conclude that *if Tweety is a bird and a penguin, we may (defeasibly) assume that Tweety does fly*?

Any consequence relation, worthy of being called "reasonable non-monotonic consequence relation", should satisfy the following six properties, known as the KLM properties, or System **P**, (here $\models$ denotes classical validity):

(REF) $\alpha \rightarrowtail \alpha$ (reflexivity),

(LLE) if $\models \alpha \leftrightarrow \beta$ and $\alpha \rightarrowtail \gamma$ then $\beta \rightarrowtail \gamma$ (left logical equivalence),

(RW) if $\models \alpha \rightarrow \beta$ and $\gamma \rightarrowtail \alpha$ then $\gamma \rightarrowtail \beta$ (right weakening),

(CUT) if $\alpha \wedge \beta \rightarrowtail \gamma$ and $\alpha \rightarrowtail \beta$ then $\alpha \rightarrowtail \gamma$,

(CM) if $\alpha \rightarrowtail \beta$ and $\alpha \rightarrowtail \gamma$ then $\alpha \wedge \gamma \rightarrowtail \beta$ (cautions monotonicity), and

(OR) if $\alpha \rightarrowtail \gamma$ and $\beta \rightarrowtail \gamma$ then $\alpha \vee \beta \rightarrowtail \gamma$.

A consequence relation that satisfies all those properties is called a preferential consequence relation. Let Δ be a set of defaults. Then,

$$\Delta \vdash_{\mathbf{P}} \alpha \rightarrowtail \beta$$

denotes that $\alpha \rightarrowtail \beta$ can be deduced from Δ using System **P**.

It has been proven ([12]) that for a ${}^*\mathbb{R}$-probabilistic model **M** following holds:

- $\mathbf{M} \models \beta \rightarrowtail_{\mathbf{M}} \alpha$ iff $\mathbf{M} \models \mathrm{CP}_{\approx 1}(\alpha, \beta)$.

This also means that satisfiability checking of a given set of defaults in System **P** can be reduced to finding a solution for a system of linear inequalities. Satisfiability problem in this logic is usually denoted by DefSAT.

3 Experimental results

Although satisifiability problem in systems described above can be reduced to solving a system of linear inequalities applying exact methods can not give satisfactory results. Number of unknowns in corresponding system of inequalities is growing exponentially with number of propositional letters in given set of formulas. Number of inequalities is, also larger then number of formulas in the set of formulas. If L is the number of formulas in the set, the number of inequalities in corresponding system is $L+1$ for logic LPP_2^{exp}, at least $2L+1$ for logic $LPCP$, and $3L+2$ for the System **P**. In case of applying an exact algorithm to those inequalities it is necessary to add restriction for atoms' measure, since measure of every atom should be greater or equal to zero. This can be avoided in heuristic approaches. Additionally, all coefficients and probability measures of atoms in a system corresponding to formulas from logic $LPCP$ or System **P** are not rational numbers, but, as previously stated, rational functions depending infinitesimal ε. This specific way of representing numbers is the reason why no commercial software can be used to solve these problems.

System **P** is considered in [8] and experimental results for applying different method for checking existence of solution for corresponding system of inequalities are presented. In most cases Fourier-Motzkin Elimination method failed. Reason for that is exponential growth of the number of inequalities during elimination of variables from the system. For example, for testing satisfiability for a set of formulas $\{bird \rightarrowtail fly, penguin \rightarrowtail bird, penguin \rightarrowtail \neg fly, fly \rightarrowtail \neg penguin\}$, the obtained system contains 30 inequalities. By eliminating the first variable, the size of the system becomes 26, after elimination of the second variable the system has 29 inequalities, then 38, 63, 229, 5494 inequalities. After eliminating seven variables, the system contains about $2 \cdot 10^6$ inequalities. An even more drastic example is testing satisfiability of the set $\{\theta \rightarrowtail \phi, \theta \rightarrowtail \psi\}$ where the number of inequalities in the system starts from 24, and then becomes 21, 30, 39, 118, 444, 37014, while after eliminating seven variables, resulting system contains about $6 \cdot 10^6$ inequalities. At this point execution stopped because there was not enough RAM.

Applying Simplex method on the same problem is also inadequate. First, it is not easy to define adequate objective function. Second, inequalities in the system can contain relations $\leq$, $>$ and $<$ which is not acceptable. Necessary transformations of the system into Simplex-acceptable form caused the wrong answers in certain situations.

Unlike the Fourier-Motzkin Elimination method and Simplex method, the application of heuristic methods gave very good results for all considered systems.The heuristic approach achieves an advantage because it does not solve the resulting system, but tries to "guess" the solution. The initial solution is generated randomly, and then it is being modified through iterations. The obtained system of inequalities is sparse, so it is relatively easy to find a solution.

All implementations are executed on a cluster that consists of 22 working nodes, each configured with two Intel 2.6GHz 8-core processors, and 64GB of memory capacity (4GB RAM per core) under Scientific Linux 6.5 64-bit with gcc version 4.4.

3.1 Logic LPP_2^{ext}

Meta-heuristics Variable Neighborhood Search (VNS) [13, 14] and Bee Colony Optimisation (BCO) [15, 16] are used and their efficiency compared. For testing purpose a number of examples is generated. All sets of formulas are satisfiable. Number of propositional letters (N) and number of formulas in the set (L) are varied ($(N, L) \in \{(50, 50), (50, 100), (50, 250), (100, 100), (100, 200), (100, 500), (200, 200), (200, 400), (200, 1000)\}$). Up to three different instances for (N, L) combination are generated.

For VNS heuristic and selected training set of formulas, the algorithm showed the best results when the maximum number of neighborhoods was set to 5 ($k_{max} = 5$), and when the local search procedure was looking for the best solution in the neighborhood of current solution. The local search looking for the first better result has a slightly lower total search time, but the success rate is much lower. Also for $k_{max} > 5$ success rate is higher, but execution time is significantly higher.

Tuning of parameters for BCO heuristic was performed on the same training set. Chosen optimal parameters were: number of bees $B = 10$, number of alternating forward and backward passes $NC = 90$, allowing degradation of objective function, and transferring some of the best results from one generation to next one.

Every instance is tested 30 times with the same setup and average execution time and success rate is calculated. Obtained results for both heuristics are given in Table 1 and Figures 1 and 2. Due to the clarity in Figure 1 the time axis is limited to 100s. As many as 6 examples using the VNS algorithm have a execution time greater than 100s, which can be seen in Table 1

N, L, $Inst$	VNS		BCO	
	Time [s]	Succ [%]	Time [s]	Succ [%]
100, 100, 1	0.15	100.00	0.01	100.00
100, 100, 2	0.55	100.00	0.08	100.00
100, 100, 3	0.01	100.00	0.01	100.00
100, 200, 1	25.02	100.00	0.31	100.00
100, 500, 1	313.60	100.00	8.73	100.00
200, 1000, 1	4689.11	100.00	29.97	100.00
200, 200, 1	4774.73	0.00	90.15	13.33
200, 200, 2	15.91	100.00	0.07	100.00
200, 400, 1	44.53	100.00	0.47	100.00
50, 100, 1	312.43	43.33	3.96	100.00
50, 100, 2	1.81	100.00	0.08	100.00
50, 100, 3	375.36	40.00	5.09	100.00
50, 250, 1	46.48	100.00	12.09	100.00
50, 250, 2	318.57	66.67	66.78	56.67
50, 250, 3	48.64	100.00	1.08	100.00
50, 50, 1	0.01	100.00	0.01	100.00
50, 50, 2	0.01	100.00	0.01	100.00
50, 50, 3	0.01	100.00	0.01	100.00

Table 1: Solving PSAT using VNS and BCO

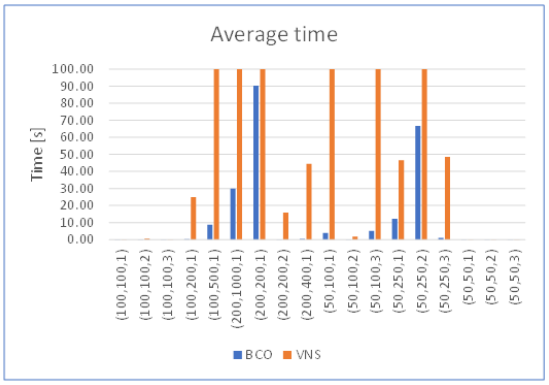

Figure 1: Average execution time for solving PSAT using VNS i BCO

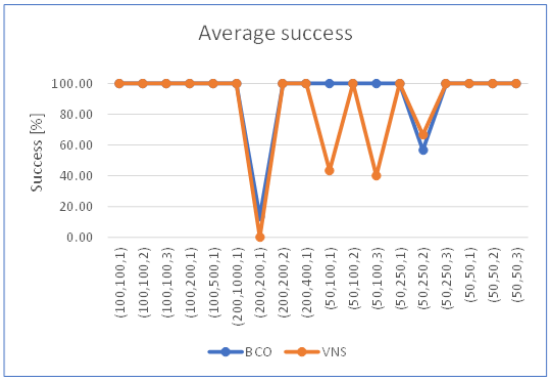

Figure 2: Average execution time for solving PSAT using VNS i BCO

It is noticeable that in only one case the BCO method had poorer success rate, while in all other cases it had better or the same. The execution time of the BCO method is significantly shorter than with the VNS method.

3.2 Logic LPCP

For the purpose of testing the possibility of applying the heuristic method on the satisfiabilty of $LPCP$ logic, satisfiable sets of formulas with a different number of propositional letters and the different numbers of formulas in the set were also created ($N \in \{10, 20, 30, 40, 50\}$, $L \in \{20, 30, 40, 50, 60, 100, 250\}$). Three different instances for each (N, L) combination were generated.

A series of tests was performed to find the optimal values of the parameters in BCO method. Execution time for these examples is significantly larger then in the previous case, so each example was tested 10 times with the same parameters values. As optimal values $B = 10$ and $NC = 30$ were selected. Maximum number of iterations is set to 50, degradation of objective function is not allowed and new generation is generated for every iteration. Results obtained with these parameters are given in the Table 2.

Most examples are successfully solved. Unsuccessful restarts significantly increase average execution time.

Table 2: BCOi results for all test examples

N, L, Inst	Succ	Time [s]
10, 30, 0	10/10	7.768
10, 30, 1	10/10	8.000
10, 30, 2	4/10	1819.987
10, 50, 0	10/10	82.972
10, 50, 1	10/10	378.609
10, 50, 2	10/10	41.336
10, 100, 0	0/10	–
10, 100, 1	10/10	208.448
10, 100, 2	10/10	14.491
20, 20, 0	10/10	3.856
20, 20, 1	10/10	16.585
20, 20, 2	10/10	59.054
20, 50, 0	10/10	44.816
20, 50, 1	10/10	19.229
20, 50, 2	9/10	10771.538
20, 100, 0	10/10	128.694
20, 100, 1	10/10	541.613
20, 100, 2	10/10	75.885
20, 250, 0	10/10	7571.212
20, 250, 1	0/10	–
20, 250, 2	0/10	–
30, 30, 0	0/10	–
30, 30, 1	10/10	18.167
30, 30, 2	10/10	53.004
30, 60, 0	10/10	153.523
30, 60, 1	10/10	408.450
30, 60, 2	10/10	74.316
30, 100, 0	0/10	–
30, 100, 1	10/10	4497.525
30, 100, 2	10/10	3793.997
30, 250, 0	10/10	442.865
30, 250, 1	10/10	3369.639
30, 250, 2	0/10	–
40, 40, 0	0/10	–
40, 40, 1	10/10	8.922
40, 40, 2	10/10	412.283
50, 50, 0	0/10	–
50, 50, 1	10/10	4118.458
50, 50, 2	3/10	23622.824

3.3 System **P**

For testing purposes we selected 18 examples from the literature [5, 6, 17], which gave us 48 satisfiable sets. Those sets are small dimension sets ($3 \leq N \leq 5$, $2 \leq L \leq 5$). BCO method showed very good results with almost 100% success rate with about 15s average execution time. Optimal BCO parameters this time were $B = 5$ and $NC = 20$. For further testing, again, satisfiable sets of formulas were generated. To improve success rate BCO parameters were increased and set to $B = 20$ and $NC = 60$. Obtained results are shown in Table 3

The obtained results in most cases show a very high success rate, but due to the extremely long execution time, it could hardly be usable in solving real-life problems. Accelerations of about 12 times were obtained by the first attempts at parallelization. There are examples in the literature in which the parallelization of the BCO method was realized by several colonies independently solving the problem on different processors and only occasionally exchanging information about the best achieved solution. Since the most time consuming part of algorithm in this problem is evaluation of objective

Table 3: Test results of large dimension examples

N, L, $Inst$	$B = 5$, $NC = 20$			$B = 20$, $NC = 60$		
	Succ	Iter. avg	Time[s]	Succ	Iter. avg	Time [s]
10, 10, 1	10/10	1.1	3.48	10/10	1.0	6.38
10, 10, 2	1/10	34.0	1282.77	10/10	3.4	1164.86
10, 10, 3	10/10	2.9	67.35	10/10	1.5	225.05
10, 10, 4	9/10	21.2	811.23	10/10	2.4	714.83
20, 20, 1	4/10	12.0	1528.01	10/10	2.5	2433.87
20, 20, 2	0/10	–	–	8/10	3.0	3162.54
20, 20, 3	10/10	1.8	110.32	10/10	1.2	348.51
20, 20, 4	0/10	–	–	10/10	8.8	12027.23
20, 50, 1	10/10	1.2	128.37	10/10	1.2	1019.38
20, 50, 2	3/10	10.3	7851.76	10/10	1.5	4875.90
20, 50, 3	10/10	1.5	395.46	10/10	1.2	1496.90
20, 50, 4	10/10	3.5	2132.46	10/10	1.2	1957.91
50, 100, 1	10/10	1.1	346.42	10/10	1.0	698.44
50, 100, 2	10/10	1.2	351.08	10/10	1.0	638.26
50, 100, 3	10/10	1.6	2069.07	10/10	1.2	6055.86
50, 100, 4	10/10	1.2	502.74	10/10	1.0	698.09
100, 100, 1	0/10	–	–			
100, 100, 2	0/10	–	–	Time for one		
100, 100, 3	0/10	–	–	iteration $\approx$ 12h		
100, 100, 4	0/10	–	–			

function it is necessary to apply different approach for parallelization.

4 Conclusions

Representing uncertain knowledge and drawing conclusions from such knowledge is an integral part of solving real-life problems. The presented systems provide a good theoretical basis for application in various fields. The results obtained by applying heuristics to the considered problems indicate the great potential of the proposed solutions.

Many directions for further research are open. It has been shown that in the case of PSAT problems the BCO method gives better results than the VNS method. The question remains how some other heuristic methods would behave. It would be specially interesting to compare the obtained results with the results of application of other heuristics to the CPSAT-ε and DefSAT problems. A different approach in parallelization for all considered problems should be tested. An additional challenge is the use of proposed systems and methods in solving real-life problems.

References

[1] Fagin R, Halpern JY, Megiddo N, A logic for reasoning about probabilities, Information and computation , 87 (1), 78 - 128, 1990.

[2] Rašković M, Classical logic with some probability operators, Publ. Inst. Math., Nouv. Sér, 53 (67), 1–3, 1993.

[3] Ognjanović Z, Rašković M, Some first-order probability logics, Theoretical Computer Science, 247 (1), 191-212, 2000.

[4] Rašković M., Marković Z., Ognjanović Z, A logic with approximate conditional probabilities that can model default reasoning, International Journal of Approximate Reasoning, 49 (1) , 52 - 66, 2008.

[5] Kraus S, Lehmann D, Magidor M, Nonmonotonic reasoning, preferential models and cumulative logics, Artificial intelligence, 44 (1), 167-207, 1990.

[6] Benferhat S, Saffiotti A, Smets P, Belief functions and default reasoning, Artificial Intelligence, 122 (1), 1-69, 2000.

[7] Stojanović T, Davidović T, Ognjanović Z, Bee colony optimization for the satisfiability problem in probabilistic logic, Applied Soft Computing, 31 (0), 339 - 347, 2015.

[8] Stojanović T, Ikodinović N, Davidović T, Ognjanović Z, Automated non-monotonic reasoning in System P, Annals of Mathematics and Artificial Intelligence, 89 (5), 471-509, 2021.

[9] Stojanović T, Kaplarević-Mališić A, Ognjanović Z, An extension of the probability logic LPP_2, Kragujevac Journal of Mathematics, 33, 45-62, 2010.

[10] Keisler HJ, Elementary calculus: an infinitesimal approach, 1986, Prindle Weber & Schmidt.

[11] Robinson A, Non-standard analysis, 1966, Studies in Logic and the Foundations of Mathematics.

[12] Lehmann D, Magidor M, What does a conditional knowledge base entail?, Artificial Intelligence, 55 (1), 1-60, 1992.

[13] Mladenović N, Hansen P, Variable neighborhood search, Computers & operations research, 24 (11), 109–1100, 1997.

[14] Hansen P, Mladenović N, Moreno Perez JA, VVariable neighbourhood search: methods and applications, Annals of Operations Research, 175(1), 367-407, 2010.

[15] Lučić P, Teodorović D, Bee system: modeling combinatorial optimizationtransportation engineering problems by swarm intelligence, in: Preprints ofthe TRISTAN IV Triennial Symposium on Transportation Analysis, Sao Miguel,Azores Islands, 441 - 445, 2001.

[16] Lučić P, Teodorović D, Transportation modeling: an artificial life approach, in: Proceedings of the 14th IEEE International Conference on Tools with ArtificialIntelligence, Washington, DC, 216 – 223, 2002.

[17] Booth R, Paris JB, A note on the rational closure of knowledge bases with both positive and negative knowledge Journal of Logic, Language and Information, 7 (2), 165 - 190, 1998.