Table 1 Compare of algorithm VolComputeWithLocalSearch and exact algorithm

表 1 VolComputeWithLocalSearch 算法与精确算法的比较

cnf	*v*	VolComputeBunches	VolComputeWithLocalSearch
20	5	5.86E+11	4.7E+11
40	15	–	1.66E+27
20	8	2.71E+18	3.99E+17
50	10	–	1.16E+21
40	5	4.02E+10	1.11E+10
40	5	3.84E+10	1.52E+10
50	8	–	4.01E+11
100	8	–	8.7E+14
50	9	–	1.26E+15
80	8	5.23E+18	3.21E+17

3.2 VolComputeWithLocalSearch算法的计算性能实验

本实验的目的主要是验证 VolComputeWithLocalSearch 算法的计算能力以及影响算法性能的因素.因为在求解#SMT 问题时需要计算#SMT 实例所对应的 CNF 范式 F 的可满足赋值,CNF 范式的难度对求解#SMT 问题有一定影响,因此在本实验中,我们选择 CNF 范式中子句和变量的比值在 4~5 之间的相变区域[18−25].本实验共分为 6 组,表 2~表 8 给出了实验的结果.在表中,*instance* 表示实例名称,P 表示实例所对应的 CNF 范式中变量个数,*cls* 表示实例所对应的 CNF 范式中子句个数,V 表示实例的维数,*LB* 表示 VolComputeWithLocalSearch 算法的计算结果(科学计数法表示),*Time* 表示 VolComputeWithLocalSearch 算法的计算耗时(单位是秒),r 表示子句个数和变量个数的比值.在表 2~表 5 的数据中,各表内子句个数和变量个数的比值不变,实例的维数增加,各表间子句个数和变量个数的比值增加.在各表中,前 7 个实例的维数即 V 值逐个增加 1,剩余的 9 个实例的维数即 V 值逐个增加 5.在表 7 的数据中,子句个数和变量个数的比值和实例的维数不变,CNF 范式的规模将不断扩大.在表 8 的数据中,子句个数和变量个数的比值和实例的规模不变,实例的维数将不断增加.

表 2 给出了 VolComputeWithLocalSearch 算法在 r=4 时的计算结果.从表中的计算结果即 *LB* 的数据上看,我们可以得出 VolComputeWithLocalSearch 算法已经可以计算出当 r=4 维数是 55 的#SMT 问题.从表中的计算时间上我们可以看出,VolComputeWithLocalSearch 算法的计算时间与维数基本上成正比关系.

Table 2 Results of algorithm VolComputeWithLocalSearch with r=4

表 2 当 r=4 时,VolComputeWithLocalSearch 算法的计算结果

Instance	*P* cls *V*	LB	Time
9_36_4	9 36 4	8.91E+08	21
9_36_5	9 36 5	4.3E+11	66
9_36_6	9 36 6	5.52E+13	71
9_36_7	9 36 7	9.83E+15	129
9_36_8	9 36 8	1.56E+18	370
9_36_9	9 36 9	1.09E+20	2 219
9_36_10	9 36 10	3.16E+21	7 199
9_36_15	9 36 15	4.89E+26	8 042
9_36_20	9 36 20	3.46E+32	5 667
9_36_25	9 36 25	4.52E+37	7 532
9_36_30	9 36 30	1.38E+39	6 168
9_36_35	9 36 35	2.12E+39	4 368
9_36_40	9 36 40	1.02E+39	5 815
9_36_45	9 36 45	2.38E+39	6 176
9_36_50	9 36 50	2.04E+39	8 853
9_36_55	9 36 55	1.02E+39	10 514

表 3 给出了 VolComputeWithLocalSearch 算法在 r=4.33 时的计算结果.表中的计算结果即 *LB* 的数据上看,我们可以得出当 r=4.33 时,VolComputeWithLocalSearch 算法已经可以计算出维数是 55 的#SMT 问题.从表中的计算时间上我们可以看出,VolComputeWithLocalSearch 算法在计算时间上具有递增趋势.

Table 3 Results of algorithm VolComputeWithLocalSearch with r=4.33

表 3 当 r=4.33 时,VolComputeWithLocalSearch 算法的计算结果

Instance	P cls V	LB	Time
9_39_4	9 39 4	1.37E+09	39
9_39_5	9 39 5	3.46E+11	62
9_39_6	9 39 6	2.62E+13	30
9_39_7	9 39 7	2.13E+15	97
9_39_8	9 39 8	2.21E+18	527
9_39_9	9 39 9	2.24E+20	3 159
9_39_10	9 39 10	3.4E+21	5 266
9_39_15	9 39 15	3.91E+26	7 106
9_39_20	9 39 20	2.31E+31	4 534
9_39_25	9 39 25	9.51E+37	7 048
9_39_30	9 39 30	7.31E+38	3 828
9_39_35	9 39 35	3.06E+39	5 580
9_39_40	9 39 40	1.36E+39	6 090
9_39_45	9 39 45	2.04E+39	6 944
9_39_50	9 39 50	1.7E+39	8 059
9_39_55	9 39 55	1.36E+39	7 848

表 4 给出了 VolComputeWithLocalSearch 算法在 r=4.67 时的计算结果.从表中的计算结果即 *LB* 的数据上看,我们可以得出当 r=4.67 时,VolComputeWithLocalSearch 算法也可以计算出维数是 55 的#SMT 问题.从表中的计算时间上我们可以看出,VolComputeWithLocalSearch 算法在计算时间上也具有递增趋势.

Table 4 Results of algorithm VolComputeWithLocalSearch with r=4.67

表 4 当 r=4.67 时,VolComputeWithLocalSearch 算法的计算结果

Instance	P cls V	LB	Time
9_42_4	9 42 4	8.99E+08	22
9_42_5	9 42 5	3.31E+11	53
9_42_6	9 42 6	1.63E+13	12
9_42_7	9 42 7	1.13E+16	98
9_42_8	9 42 8	5.12E+17	199
9_42_9	9 42 9	1.18E+20	2 126
9_42_10	9 42 10	3.47E+21	6 032
9_42_15	9 42 15	7.24E+26	8 281
9_42_20	9 42 20	3.12E+32	4 738
9_42_25	9 42 25	7.36E+36	3 546
9_42_30	9 42 30	1.71E+39	4 964
9_42_35	9 42 35	3.81E+39	5 908
9_42_40	9 42 40	3.4E+39	7 503
9_42_45	9 42 45	2.38E+39	6 231
9_42_50	9 42 50	1.7E+39	4 418
9_42_55	9 42 55	1.7E+39	6 960

表 5 给出了 VolComputeWithLocalSearch 算法在 r=5 时的计算结果.从表中的计算结果即 *LB* 的数据上看,我们可以得出当 r=5 时,VolComputeWithLocalSearch 算法也可以计算出维数是 55 的#SMT 问题.从表中的计算时间上我们可以看出,VolComputeWithLocalSearch 算法在计算时间上也具有递增趋势.

Table 5 Results of algorithm VolComputeWithLocalSearch with r=5

表 5 当 r=5 时 VolComputeWithLocalSearch 算法的计算结果

Instance	P cls V	LB	Time
9_45_4	9 45 4	2.63E+08	21
9_45_5	9 45 5	1.24E+11	42
9_45_6	9 45 6	4.27E+13	79
9_45_7	9 45 7	7.45E+15	99
9_45_8	9 45 8	4.46E+17	196
9_45_9	9 45 9	7.2E+19	1 189
9_45_10	9 45 10	3.08E+21	4 949
9_45_15	9 45 15	2.52E+26	5 776
9_45_20	9 45 20	1.05E+33	4 795
9_45_25	9 45 25	4.35E+38	4 647
9_45_30	9 45 30	1.77E+39	4 232

Table 5 Results of algorithm VolComputeWithLocalSearch with r=5 (Continued)

表 5 当 r=5 时 VolComputeWithLocalSearch 算法的计算结果(续)

Instance	P cls V	LB	Time
9_45_35	9 45 35	2.72E+39	4 308
9_45_40	9 45 40	2.04E+39	6 499
9_45_45	9 45 45	2.72E+39	6 656
9_45_50	9 45 50	1.7E+39	4 393
9_45_55	9 45 55	1.7E+39	6 468

表 6 给出了表 2~表 5 的执行时间横向对比.

Table 6 Compare of run time from Table 2 to Table 5

表 6 表 2~表 5 的运行时间横向对比

P	V	r=4	r=4.33	r=4.67	r=5
9	4	21	39	22	21
9	5	66	62	53	42
9	6	71	30	12	79
9	7	129	97	98	99
9	8	370	527	199	196
9	9	2 219	3 159	2 126	1 189
9	10	7 199	5 266	6 032	4 949
9	15	8 042	7 106	8 281	5 776
9	20	5 667	4 534	4 738	4 795
9	25	7 532	7 048	3 546	4 647
9	30	6 168	3 828	4 964	4 232
9	35	4 368	5 580	5 908	4 308
9	40	5 815	6 090	7 503	6 499
9	45	6 176	6 944	6 231	6 656
9	50	8 853	8 059	4 418	4 393
9	55	10 514	7 848	6 960	6 468

从表中可以看出:当实例的维数为 9 时,VolComputeWithLocalSearch 算法的执行时间产生了较大的波动.从整体上看,VolComputeWithLocalSearch 算法在 r=4 时执行时间最长,在某种程度上也说明了#SMT 问题在 r=4 时较难求解.表 7 给出了当 V=50 时,VolComputeWithLocalSearch 算法的计算结果.从表中可以看出:随着实例规模的增加,算法的执行时间出现了递减-不变-递增的趋势.同时也可以发现,算法可以计算出维数为 50 的实例.

Table 7 Results of algorithm VolComputeWithLocalSearch with V=50

表 7 当 V=50 时 VolComputeWithLocalSearch 算法的计算结果

Instance	P cls V	LB	Time
10_42_50	10 42 50	3.4E+38	2 947
11_46_50	11 46 50	1.63E+19	1 988
12_50_50	12 50 50	1.39E+24	348
13_54_50	13 54 50	1.44E+19	996
14_59_50	14 59 50	2.1E+22	798
15_63_50	15 63 50	9.35E+16	955
16_69_50	16 69 50	1.41E+09	935
17_73_50	17 73 50	1.54E+13	1 754
18_78_50	18 78 50	4.88E+20	3 761
19_85_50	19_85_50	1.77E+13	3 632

表 8 给出了当 P=15,r=4.27 时,VolComputeWithLocalSearch 算法的计算结果.在表中所有实例的维数即 V 值逐个增加 5.从表中的计算结果即 LB 的数据上看,我们可以得出 VolComputeWithLocalSearch 算法已经可以计算出维数是 50 的#SMT 问题.从表中的计算时间上可以看出,VolComputeWithLocalSearch 算法的计算时间与维数基本上成正比关系.

Table 8 Results of algorithm VolComputeWithLocalSearch with P=15, r=4.27

表 8 当 P=15,r=4.27 时,VolComputeWithLocalSearch 算法的计算结果

Instance	P cls V	LB	Time
15_64_5	15 64 5	8.76E+08	4
15_64_10	15 64 10	2.5E+16	187
15_64_15	15 64 15	1.49E+15	299
15_64_20	15 64 20	2.78E+30	424
15_64_25	15 64 25	2.71E+25	316
15_64_30	15 64 30	6.19E+19	548
15_64_35	15 64 35	2.35E+20	848
15_64_40	15 64 40	3.4E+38	3 319
15_64_45	15 64 45	3.81E+18	858
15_64_50	15 64 50	3.46E+15	1 005

综上所述,VolComputeWithLocalSearch 算法能够处理较大规模的问题,且算法的计算耗时与实例的维数具有较强的关系,与子句数和变量个数关系不大.通过表 2~表 5 中的数据可以初步得出,算法的计算耗时与实例的维数相关的.表 7 中数据则在很大程度上排除了与实例的子句数和变量个数的关系.表 8 中数据进一步验证了算法的计算耗时与实例的维数具有较强的关系,与子句数和变量个数关系不大.

从计算结果上,VolComputeWithLocalSearch 算法的计算能力是可以到 55 维的.

3.3 VolComputeWithLocalSearch算法的精确度实验

本实验的目的是考察 VolComputeWithLocalSearch 算法的精确度.在实验中,将进行基于蒙特卡罗的体积估算算法 DirectMonteCarloMethod 的对比实验[26].基于蒙特卡罗的体积估算算法 DirectMonteCarloMethod 是一种经典的体积计算算法,该算法利用蒙特卡洛法随机均匀地在约束范围内生成顶点,然后随机取一定数量的单位球体,在球体计算顶点的密度,再利用密度估算体积.我们在实验中所用到的实例均出自于文献[26],表 9 给出了 VolComputeWithLocalSearch 算法与 DirectMonteCarloMethod 算法的比较结果.在表中,*Vol* 表示实例的实际体积(采用数学方法计算),*Time* 表示计算耗时,*Ratio* 表示精确度(算法计算的结果与精确体积的比值).从表中可以看出,VolComputeWithLocalSearch 算法的精确度都达到了 95%以上.

在与算法 DirectMonteCarloMethod 精确度的比较上,VolComputeWithLocalSearch 算法除了在 Sample1, Sample5,Sample6 略低于算法 DirectMonteCarloMethod,在其他实例上均有较大幅度的提高.尤其对于实例 Sample3,精确度从原来的 0.01777 提高到 0.95321.在计算耗时方面,VolComputeWithLocalSearch 算法则全面优胜于 DirectMonteCarloMethod 算法,最大的提高幅度约达 95.8%(实例 Sample8).

Table 9 Compare of accuracy of algorithm VolComputeWithLocalSearch and DirectMonteCarloMethod

表 9 VolComputeWithLocalSearch 与 DirectMonteCarloMethod 在精确度方面的比较结果

	Vol	DirectMonteCarloMethod		VolComputeWithLocalSearch	
		Ratio	Time	Ratio	Time
Sample1	253	**0.998 54**	873.838	0.993 50	**190.891**
Sample2	20 000	0.642 20	1 882.048	**0.999 68**	**159.683**
Sample3	400	0.011 06	643.291	**0.994 37**	**95.292**
Sample4	20 000	0.998 02	1 664.748	**0.999 70**	**89.452**
Sample5	1 600 000 000	**0.996 26**	1 963.366	0.960 01	**164.553**
Sample6	2763562.5	**0.999 19**	2 039.159	0.993 60	**174.18**
Sample7	235 200 000	0.017 77	2 135.570	**0.953 21**	**172.027**
Sample8	12916.66699	0.443 85	2 028.618	**0.999 54**	**85.446**

3.4 VolComputeWithLocalSearch算法的稳定性实验

本实验的目的是考察 VolComputeWithLocalSearch 算法的稳定性.在实验中,同样与基于蒙特卡罗的体积估算算法 DirectMonteCarloMethod[26]进行比较.我们在实验中所用到的实例均出自于文献[26],每个实例进行 10 次重复实验,分别记录 VolComputeWithLocalSearch 算法和 DirectMonteCarloMethod 算法的计算结果和执行时间.表 10、表 11 给出了 VolComputeWithLocalSearch 算法与 DirectMonteCarloMethod 算法的稳定性实验结果,

其中,*Ratio* 为算法的精确度(计算结果与真实值的比值),*Time* 表示耗时,*Average* 是精确度的平均值,*Variance* 是精确度的方差.

Table 10 Results of stability of algorithm VolComputeWithLocalSearch

表 10 VolComputeWithLocalSearch 算法稳定性实验结果

实验次数		Sample1	Sample2	Sample3	Sample4	Sample5	Sample6	Sample7	Sample8
1	Ratio	0.993 518	0.999 58	0.994 518	0.999 815	0.960 5	0.994 709	0.951 577	0.999 569
	Time	192.20	161.52	91.96	89.07	161.34	172.74	166.53	87.68
2	Ratio	0.993 494	0.999 7	0.994 523	0.999 645	0.962 844	0.992 422	0.949 345	0.999 569
	Time	190.7	157.50	96.61	89.66	165.43	178.04	173.68	85.87
3	Ratio	0.994 538	0.999 685	0.995 19	0.999 675	0.962 506	0.992 067	0.954 528	0.999 515
	Time	189.11	158.98	91.81	89.73	164.40	174.70	173.63	85.58
4	Ratio	0.993 96	0.999 555	0.995 508	0.999 685	0.956 594	0.995 034	0.952 696	0.999 445
	Time	190.53	161.18	94.51	89.93	166.91	174.23	174.44	86.6
5	Ratio	0.993 174	0.999 725	0.993 428	0.999 645	0.958 85	0.992 871	0.952 636	0.999 569
	Time	191.05	158.89	96.54	89.93	165.10	173.43	175.70	82.00
6	Ratio	0.992 767	0.999 665	0.992 558	0.999 725	0.958 756	0.992 194	0.958 95	0.999 639
	Time	190.43	157.02	98.55	90.40	167.39	173.68	171.73	85.51
7	Ratio	0.994 012	0.999 715	0.993 93	0.999 675	0.959 756	0.995 404	0.955 48	0.999 561
	Time	191.13	159.95	99.29	88.50	163.20	174.10	173.19	85.18
8	Ratio	0.993 549	0.999 635	0.993 703	0.999 745	0.960 388	0.993 877	0.948 508	0.999 561
	Time	190.1	158.79	97.37	89.03	164.63	174.57	174.18	85.54
9	Ratio	0.993 791	0.999 705	0.994 965	0.999 72	0.959 944	0.992 831	0.955 901	0.999 554
	Time	192.42	159.32	91.66	89.26	164.01	171.74	170.07	84.43
10	Ratio	0.992 277	0.999 835	0.995 458	0.999 69	0.960 044	0.994 676	0.952 483	0.999 507
	Time	191.24	163.68	94.62	89.01	163.12	174.57	167.12	86.07
Average	Ratio	0.993 508	0.999 68	0.994 378	0.999 702	0.960 018	0.993 608	0.953 21	0.999 549
	Time	190.89	159.68	95.29	89.45	164.55	174.18	172.03	85.44
Variance	Ratio	3.78E−07	5.62E−09	8.36E−07	2.39E−09	2.93E−06	1.46E−06	8.79E−06	2.33E−09
	Time	0.843 249	3.557 221	7.169 236	0.293 876	2.920 401	2.432 28	8.881 721	1.990 044

Table 11 Results of stability of algorithmDirectMonteCarloMethod

表 11 DirectMonteCarloMethod 算法稳定性实验结果

实验次数		Sample1	Sample2	Sample3	Sample4	Sample5	Sample6	Sample7	Sample8
1	Ratio	0.992 672	0.640 181	0.010 35	0.997 055	1.006 677	0.999 792	0.018 01	0.441 733
	Time	862.966 8	1 884.131	650.426 8	1 656.687	1 950.42	1 996.876	2 138.781	2 015.006
2	Ratio	1.009 394	0.630 025	0.010 828	1.001 698	1.000 717	0.999 079	0.017 349	0.445 468
	Time	869.329 6	1 892.382	674.389 2	1 655.636	1 955.599	2 073.645	2 129.395	2 023.088
3	Ratio	0.997 644	0.638 916	0.010 958	0.999 352	1.002 781	0.999 936	0.016 612	0.446 69
	Time	896.416 6	1 880.987	658.370 1	1 658.089	1 957.85	2 064.349	2 147.92	2 032.872
4	Ratio	1.007 667	0.643 082	0.010 517	1.000 008	1.001 123	0.990 397	0.017 375	0.446 845
	Time	858.967 4	1 884.524	639.438 5	1 661.594	1 958.3	2 026.864	2 128.407	2 034.148
5	Ratio	1.006 079	0.650 907	0.010 847	0.997 399	1.002 125	1.004 173	0.017 715	0.438 905
	Time	882.418 6	1 886.095	637.982 2	1 658.089	1 967.531	2 019.067	2 134.088	2 032.021
6	Ratio	0.996 28	0.639 319	0.011 088	1.003 02	0.992 118	0.999 423	0.016 46	0.437 729
	Time	883.872 9	1 882.952	636.393 5	1 658.79	1 970.908	2 023.865	2 139.028	2 030.745
7	Ratio	0.997 893	0.625 419	0.010 943	1.000 962	0.999 582	0.996 836	0.017 539	0.434 64
	Time	880.418 9	1 879.415	632.024 7	1 661.243	1 966.856	2 068.547	2 131.371	2 032.872
8	Ratio	0.997 74	0.637 704	0.010 812	1.000 594	0.990 889	0.999 735	0.017 859	0.445 648
	Time	857.876 7	1 877.451	632.554 3	1 659.491	1 970.458	2 024.165	2 138.781	2 028.618
9	Ratio	0.994 025	0.632 085	0.010 733	0.998 887	0.997 974	1.005 272	0.016 393	0.461 062
	Time	874.783 3	1 874.7	634.804 9	1 695.94	1 967.081	2 068.547	2 125.69	2 029.894
10	Ratio	1.010 849	0.643 383	0.011 403	0.997 625	0.991 023	0.997 427	0.017 717	0.431 717
	Time	871.329 3	1 877.844	636.525 9	1 681.921	1 968.657	2 025.665	2 142.239	2 026.916
Average	Ratio	1.001 024	0.638 102	0.010 848	0.999 66	0.998 501	0.999 207	0.017 303	0.443 044
	Time	873.838	1 882.048	643.291	1 664.748	1 963.366	2 039.159	2 135.57	2 028.618
Variance	Ratio	4.1E−05	4.86E−05	7.59E−08	3.49E−06	2.66E−05	1.49E−05	3.23E−07	6.11E−05
	Time	134.902	23.531 97	169.598 2	158.993 1	46.251 82	653.454 9	43.646 17	30.326 33

可以看出:

- 在算法精确度的平均值上,相较于 DirectMonteCarloMethod 算法,VolComputeWithLocalSearch 算法计

算用时均较短;

- 从算法精确度的方差来看,VolComputeWithLocalSearch 算法除了在实例 Sample3 和 Sample7 的精确度的方差较大以外,其他实例不论在精确度还是在时间上都全面优于 DirectMonteCarloMethod 算法.

综上所述,VolComputeWithLocalSearch 算法在收敛速度上变化不大,算法在迭代次数上是稳定的,因此,VolComputeWithLocalSearch 算法是稳定的.

综合以上数据分析,VolComputeWithLocalSearch 算法在小规模问题上精确度和计算能力上是有保证的;在大规模问题上,由于目前还没有精确算法能够与之相比较,故在精确度等方面有待考证.通过实验,我们的算法计算耗时与维数成不严格的正比例关系,与实例的子句数和变量个数关系不大.由此可见,#SMT 的求解效率的瓶颈并不是 SAT 求解器,而是在于体积计算过程.

4 结论和展望

本文设计了一种求解 SMT 计数问题的近似求解器,在现有的#SMT 精确求解算法的基础上加入差分进化算法,提出了一种新的局部搜索算法 VolComputeWithLocalSearch,通过调用体积计算工具 qhull,进而给出#SMT 问题的近似解.我们从理论上证明了 VolComputeWithLocalSearch 算法可以得到精确解的下界.实验结果表明:VolComputeWithLocalSearch 算法是稳定的,具有快速的求解能力,并在高维问题上具有很好的表现.

对于基于线性公式的背景理论的#SMT 问题且只要求下界的问题应用中,本文的研究可以提供很大的帮助,但 VolComputeWithLocalSearch 算法仅可以求解基于线性公式的背景理论的#SMT 实例,对于其他背景理论如非线性公式、位向量、指针等理论的求解则还有待突破.

References:

[1] Cook SA. The complexity of theorem-proving procedures. In: Proc. of the 3rd Annual ACM Symp. on Theory of Computing. ACM Press, 1971. 151−158. [doi: 10.1145/800157.805047]

[2] Li J, Liu WF. A survey on theoretical combination techniques of SMT solvers. Computer Engineering & Science, 2011,33(10): 111−119 (in Chinese with English abstract).

[3] Bozzano M, Bruttomesso R, Cimatti A, Junttila T, Ranise S, van Rossum P, Sebastiani R. Efficient satisfiability modulo theories via delayed theory combination. In: Proc. of the Computer Aided Verification. Berlin Heidelberg: Springer-Valag, 2005. 335−349. [doi: 10. 1007/11513988_34]

[4] Nieuwenhuis R, Oliveras A, Tinelli C. Solving SAT and SAT modulo theories: From an abstract davis-putnam-logemann-loveland procedure to DPLL(*T*). Journal of the ACM, 2006,53(6):937−977. [doi: 10.1145/1217856.1217859]

[5] Jha S, Limaye R, Seshia SA. Beaver: Engineering an efficient smt solver for bit-vector arithmetic. In: Proc. of the Computer Aided Verification. Berlin, Heidelberg: Springer-Verlag, 2009. 668−674. [doi: 10.1007/978-3-642-02658-4_53]

[6] Zhou J, Su WH, Wang JY. New worst-case upper bound for counting exact satisfiability. Int'l Journal of Foundations of Computer Science, 2014, 25(6):667−678.

[7] Huang P, Yin MH, Xu K. Exact phase transitions and approximate algorithm of #CSP. In: Proc. of the AAAI. 2011.

[8] Ma FF, Liu S, Zhang J. Volume computation for boolean combination of linear arithmetic constraints. In: Proc. of the Automated Deduction. Springer-Verlag, 2009. 453−468. [doi: 10.1007/978-3-642-02959-2_33]

[9] Gomes CP, Hoffmann J, Sabharwal A, Selman B. From sampling to model counting. In: Proc. of the 20th Int'l Joint Conf. on ArtificialIntelligence (IJCAI 2007). 2007. 2293−2299.

[10] Dyer ME, Frieze AM. On the complexity of computing the volume of a polyhedron. SIAM Journal on Computing, 1988,17(5): 967−974. [doi: 10.1137/0217060]

[11] Davis M, Putnam H. A computing procedure for quantification theory. Journal of the ACM, 1960,7(3):201−215. [doi: 10.1145/321033.321034]

[12] Davis M, Logemann G, Loveland D. A machine program for theorem-proving. Communications of the ACM, 1962,5(7):394−397. [doi: 10.1145/368273.368557]

[13] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997,11(4):341−359. [doi: 10.1023/A:1008202821328]

[14] Storn R. On the usage of differential evolution for function optimization. In: Proc. of the '96 Biennial Conf. of the North American Fuzzy Information Processing Society (NAFIPS). IEEE, 1996. 519−523. [doi: 10.1109/NAFIPS.1996.534789]

[15] Storn R, Price K. Minimizing the real functions of the ICEC'96 contest by differential evolution. In: Proc. of the Int'l Conf. on Evolutionary Computation. 1996. 842−844. [doi: 10.1109/ICEC.1996.542711]

[16] Li XT, Yin MH. Application of differential evolution algorithm on self-potential data. PloS one, 2012,7(12):e51199. [doi: 10.1371/journal.pone.0051199]

[17] Li XT, Yin MH. An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. Advances in Engineering Software, 2013,55:10−31. [doi: 10.1016/j.advengsoft.2012.09.003]

[18] Zhou JP, Yin MH, Zhou CG. New worst-case upper bound for #2-SAT and #3-SAT with the number of clauses as the parameter. In: Proc. of the 24th AAAI Conf. on Artificial Intelligence (AAAI 2010). 2010. 217−222.

[19] Gao J, Wang JN, Yin MH. Experimental analyses on phase transitions in compiling satisfiability problems. Science China Information Sciences, 2015,58(3):1−11. [doi: 10.1007/s11432-014-5154-0]

[20] Huang P, Yin MH. An upper (lower) bound for max (min) CSP. Science China Information Sciences, 2014,57(7):1−9. [doi: 10.1007/s11432-013-5052-x]

[21] Zhou JP, Yin MH, Gu WX, Sun JG. Research on decreasing observation variables for strong planning under partial observation. Ruan Jian Xue Bao/Journal of Software, 2009,20(2): 290−304 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3152.htm [doi: 10.3724/SP.J.1001.2009.03152]

[22] Yin MH, Sun JG, Lin H, Wu X. Possibilistic extension rules for reasoning and knowledge compilation. Ruan Jian Xue Bao/Journal of Software, 2010,21(11):2826−2837 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3690.htm [doi: 10.3724/SP.J.1001.2010.03690]

[23] Yin MH, Zhou JP, Sun JG, Gu WX. Heuristic survey propagation algorithm for solving QBF problem. Ruan Jian Xue Bao/Journal of Software, 2011,22(7):1538−1550 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3859.htm [doi: 10.3724/SP.J.1001.2011.03859]

[24] Yin MH, Lin H, Sun JG. Solving #SAT using extension rules. Ruan Jian Xue Bao/Journal of Software, 2009,20(7):1714−1725 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3320.htm [doi: 10.3724/SP.J.1001.2009.03320]

[25] Xu L. Improved SMT-Based Bounded Model Checking for Real-Time Systems. Ruan Jian Xue Bao/Journal of Software, 2010, 21(7):1491−1502. http://www.jos.org.cn/1000-9825/585.htm [doi: 10.3724/SP.J.1001.2010.00585]

[26] Liu S, Zhang J, Zhu B. Volume computation using a direct Monte Carlo method. In: Proc. of the Computing and Combinatorics. Springer-Verlag, 2007. 198−209. [doi: 10.1007/978-3-540-73545-8_21]

附中文参考文献:

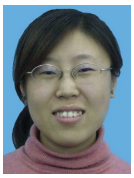
[2] 李婧,刘万伟.SMT 求解器理论组合技术研究.计算机工程与科学,2011,33(10):111−119.

[21] 周俊萍,殷明浩,谷文祥,孙吉贵.部分可观察强规划中约减观察变量的研究.软件学报,2009,20(2):290−304. http://www.jos.org.cn/1000-9825/3152.htm [doi: 10.3724/SP.J.1001.2009.03152]

[22] 殷明浩,孙吉贵,林海,吴瑕.可能性扩展规则的推理和知识编译.软件学报,2010,21(11):2826−2837. http://www.jos.org.cn/1000-9825/3690.htm [doi: 10.3724/SP.J.1001.2010.03690]

[23] 殷明浩,周俊萍,孙吉贵,谷文祥.求解 QBF 问题的启发式调查传播算法.软件学报,2011,22(7):1538−1550. http://www.jos.org.cn/1000-9825/3859.htm [doi: 10.3724/SP.J.1001.2011.03859]

[24] 殷明浩,林海,孙吉贵.一种基于扩展规则的#SAT 求解系统.软件学报,2009,20(7):1714−1725. http://www.jos.org.cn/1000-9825/3320.htm [doi: 10.3724/SP.J.1001.2009.03320]



周俊萍(1981−),女,吉林蛟河人,博士,讲师,CCF 会员,主要研究领域为智能规划,自动推理.

曾志勇(1989−),男,硕士生,主要研究领域为并行计算,大数据处理.

李睿智(1989−),女,博士生,主要研究领域为算法设计与分析.

殷明浩(1979−),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为自动推理,智能规划.