Report inde

StatProfilerHTML.jl report

otati romeri		Generated on Tue 31 May 2022 03:17:20 PM -0 File source code
ine Exclusive Incl	clusive	File Source Code  Code  function classic@Pieq(NMMdata::NMRType,
!		c::Floats4, virtual_c::Floats4,
3 1 5		allmol :: 8ool,maxl=25,bug=true) if allmol := true
7		error("This solver is not prepared to find all solutions yet")  end
		n = NORGAIR.din
10		nol = Molecula type (Vector (Atom) yee) (underf, $n$ ), $\theta$ . $\theta$ )  for $i=1:n$
1 2		nol.atons[1] = AtonType(0.0,0.0,0.0) end
13		nsol = 0 storage_mol = Dict(Int54,MoleculeType)()
15 16		e.g.mon = 0ngmn = 0ngm
17		#count_mpp = [*,*,*,!,*] reg_note = [*,*,*,!,*]
19		rep_off = [1,0,8,0] rep_opth = [6,8,6,8]
21 22		# first atom mol. atoms[1].element = NMEdata.info[1,:].naval[1].atom1
23		nol. atom(1), $x = 0.8$ nol. atom(1), $y = 0.8$
25 26		nol. atoms [1], $z=0.8$ free cond atom
27		nol.atoms[2], element = NMMdata.info[2],:].nxval[1].atoml nol.atoms[2], x = NMMdata.info[1],2].dist
29		sol. stors[2], $y = 0.0$ sol. stors[2], $z = 0.0$
31		# tird atom
32		012 = NMPGdx13.Info[1,2].dist 013 = NMPGdx13.Info[1,3].dist
34 35		D23 = NMRdsta.info[2,3].dist D14 = 0.0
36 37		024 = 0,0 034 = 0,0
38 39		c6,16 = bondangle (012,013,023)  cop_node == [3,6,1,1]
40 41		cu, xu = (0.0,0.0) sol.atoms[3].olement = NMMdata.info[3,:].nxval[1].atom1
12 13		nol. stors[3] x = -012-023*r8 nol. stors[3] y = 023*s8
14		nol.atons[3], z = 0.8 nop.node == [1,2,0,0]
46 47		C = zero(4,4) C(1,4) = nol.zton(3),x
47 48 49		C[3,4] = nol.atom(3].y  (3,4] = nol.atom(3].z
50		C(1,1) = c0
52		C(1,2) = 18 C(2,1) = 18
53 54		C(2,2) = -c8 C(3,3) = -1.0
56		C(4,4) = 1.0
57 58		1 = 4 # branching starts at atom 4 pos = 4 # position in virtual path
59 60		esplore_right_tide = zero(thod.no) Clist = Array(tary)floatefs_1D(condr.no) = to access level 1 it is reced to put 1-3
61 62		# println(C)  C_List[3] = copy(C)
63 64		# C_before = zeros(4,4)  B = Array(Array(Flont64,2))(undef,n) # to access level l it is need to put 1-3
65 66		8(4) = 2002(4,4) first_occ = -1*ones[Int,0)
67 68		virtual_comt = 0 while lcom M D-3
59		Fif 4cl co starl
70		# println("8 matrix in level \$(1-1) ") # display(8(1-1))
72		<pre># println("C astrix") # display(c_list([-1])</pre>
74		# # # # # # # # # # # # # # # # # # #
76		etigalay(i)
78 79		if lemand: error(bab*)
B0 B1		end  s if first_cos([1:1] = -1 Ms first_cos([WM6sta.virtus]_psth[1:1]) == 1
B2 B3		# first_corr[1-1] = NMMdata.virtual_path[1-1] # end
B4 B5		# T000: otisizar!  pos = findfirst[x->xv=l-1,[WMSdsta.virtual_path) +1
86 87		#E_before = zeros(4,4)  C_before = copy(_List[1-1])
88		kep = trus
89 90		while keep  try  Old = NR-data (efc1NR-data pirtual earblook 21 NR-data pirtual earblook 21 distriction)
91		D14 = NMRdsta.info[NMRdsta.virtual_path[pos-3],NMRdsta.virtual_path[pos]].dist catch
94		error("5[[WMGsta:virtual_path[pos:3],WMGsta:virtual_path[pos:3]].")  D14 = sqrt((mol.stoss[[WMGsta:virtual_path[pos:3]].x - mol.stoss[[WMGsta:virtual_path[pos:3]].x - mol.stoss[[WMGsta:virtual_path[pos:3]].y - mol.stoss[[WMGsta:virtual_path[pos:3]].y - mol.stoss[[WMGsta:virtual_path[pos:3]].x - mol.stoss[[WMGsta:virtual_path
16		end try
17		024 = NMEdata.info[NMEdata.virtual_path[pos-2],NMEdata.virtual_path[pos]].dist catch
100		224 = sqrt((nol.stons:[WMRdsta.virtual_path[pos-2]].x - mol.stons:[WMRdsta.virtual_path[pos-1].x)*2 + (mol.stons:[WMRdsta.virtual_path[pos-2]].y - mol.stons:[WMRdsta.virtual_path[pos-2]].z - mol.sto
101		try  1034 = NMMdata.info[NMMdata.virtual_path[pos-1], NMMdata.virtual_path[pos]] dist
103		catch  534 = Sqrt([mol.stons[WMdsta.virtual_path[pos-1]].x - mol.stons[WMdsta.virtual_path[pos]].s)*2 + (mol.stons[WMdsta.virtual_path[pos-1]].y - mol.stons[WMdsta.virtual_path[pos-1]].z - mol.stons[WMdsta.virtual_path[pos]].s)*2)
105		end try
107		U.y D12 = WMRdsta.info[WMRdsta.virtual_path[pos-2],WMRdsta.virtual_path[pos-2]].dist Gatch
109		D12 = sqrt((mol. stose [NMRdsta. virtual_path[pos-3]]; x - mol. stoses [NMRdsta. virtual_path[pos-2]]; x) = (mol. stoses [NMRdsta. virtual_path[pos-2]]; y) - mol. stoses [NMRdsta. virtual_path[pos-2]]; y) - mol. stoses [NMRdsta. virtual_path[pos-2]]; x) - mol. stoses [NMRdsta. virt
110		end try
112		013 = NMRdsta.info[NMRdsta.virtual_path[pos-3],NMRdsta.virtual_path[pos-1]].dist catch
114		013 = sqrt((sol. atoms[WMdata.virtual_path[por-3]].x - sol. atoms[WMdata.virtual_path[por-1]].x)*2 + (sol. atoms[WMdata.virtual_path[por-3]].y - sol. atoms[WMdata.virtual_path[por-1]].y)*2+ (sol. atoms[WMdata.virtual_path[por-3]].x - sol. a
116		try  023 = WMRdsta.info[WMRdsta.virtual_path[pos-2],WMRdsta.virtual_path[pos-1]].dist
118		catch  621 *qrt([sol.zens MMdsta.virtual_path[gos-2]], x * ml.zens MMdsta.virtual_path[gos-1]], x)* 2* (mol.zens MMdsta.virtual_path[gos-2]], y * ml.zens MMdsta.virtual_path[gos-1]], y)*2* (ml.zens MMdsta.virtual_path[gos-2]], x * ml.zens MMd
120		end c8, s8 = bondarg1e(023,024,034)
122		cu, we = badforsitomsq[e(12,0)3_014,023_004,024)  # println["l value = \$(1) and NMRdstavalue = \$(NMRdsta.virtual_path[pos]) in position \$(pos)")
124		# (Schow cd, sd,cu,su, 034
126		8[1] = torsiomatrix(c0,s0,cu,u034)
127		s if explore_right_tide[1] == true
129		# S[1] = torionmatrix(S[1])  # end  (A bandwinter (struct problem)
131		ff 1-000min virtual_path(pos)  op_opdor = (0.7.0,2) firstion matrix
132		non_mond = 1 (s,t_sup_ version manual non_mond = 1 (s,t_sup_ version m

6	C_list[l] = prodmatrix(C_before,8[l])
	nop_rode += [24,33,8,8] keep = false
else	nop_mode += [0.7,0.8] #formion matrix
	mp_mode = [2,5,1,1] * bond angle mp_mode = [1,5,1,1] * bond angle
	nog_upath += [0,7,0,0] # torsion matrix
	nop_upath = [3,6,1,1] # bord angle nop_upath = [18,28,4,2] # bod torsion angle
	cpx = mol.atoms[NMMdata.virtual_path[pos1]:x
	cy = nol. atons([Wedata virtual_path[nos]] y  cpr = nol. atons([Wedata virtual_path[nos]] : 2
	If bug println("{Scpu , Scpy , Scpz }") ond
١	*** **********************************
	nop_upath = [24,33,8,49] nop_node = [23,33,8,4]
	if sqrt[(Virtual_Torsion[1,4]- cps)/2=(Virtual_Torsion[2,4]- cps)/2=(Virtual_Torsion[3,4]- cps)/2]> virtual_c
	8[1] = torsiomatrix(8[1])  C_before = productrix(C_before,8[1])
	nop_vpath = [24,33,0,8]  nop_vpath = [24,33,0,8]  ### println[prison 1"]
e	### printin(*passou 2*)
	C_before = Copy(Virtual_Torsion) #C_before = Virtual_Torsion
	ond If bug
	println("Virtual Torsion") display(Virtual_Torsion)
	end  #println "Torsion matrix \$[B[1]]")
	Appintaln("Lefore matrix S(C,Defore)")  ###Bebog "virtual atom position " C,Defore[3,4],C,Defore[3,4]  pos = pos*1
end end	
if bu	_ right_side(]] == false  19 Ma (St-ken maxi)
d	println(" satrix in level 5(1) in left side") display(8(1)) println(" satrix ")
	display(C_list[i])
mol.a	tons[], element = MMGdata.info[[,:].naval[[],atonl trons[]].x = C_list[l][1,4]
mol.a	toms[1], y = Clist[1][2,4] toms[1], z = Clist[1][3,4]
count λ, co	t = [0,0,0,0]  count = pruningtest(mo),1,0MMdata,r,count)
#prin	idf == count tln['C = C_before*8 at level 5(1) left side \$(C_list[1])) = \$(C_list[1-1]) * \$(8[1])')
if $\lambda$	== 1  f lon  # println("Fartial solution by left side at level \$(1) ", nol)
	* printing factions solution by zero since at sever s(i) , most  (_branch ==1  £lasting[closure[=1,pos=1,mol.C])
	<pre>feeplore_right_side([+1]=false else</pre>
	nsol-mol·1  storage_mol[nol] = copyinol)
	@debug "Bank n was reached, a solution was found " and
	n_prose == 1 eoplor_right_side[] = true
end end	
	e_right_side[1] == true
#nop_	= torsionatrix(8[1]) pode *= [0.0,0.0]
if bu	ti[] = prodmatix(C.before,8[1])# tembo que otimizar este calculo ng Ma (3-1-cmaxl)
	println('8 matrix in level \$(1) in right side') display(8(1))
	printle('C matrix') display(c_list[1])
	node == [24,33,8,8] tros([1],x = C_list([1])[4]
mol.a	
count	= (0,0,0,0) unt = pruningtes(mo),1,NMRdsta,c.count) #preciso modificar
nop_di #prin:	66f == count wtln('C = Chefora'8 at level \$(1) right side   \$(Clist[1]) = \$(Clist[1-1]) * \$(8[1])')
if p	if Isa
	# println("Partial solution by right side at level \$(1) ", mol)  n_branch == 1  n_branch == 1
	#explore_right_tide( 1-1 =false #flastide_closure( 1-1,pos-1,mol.C) else
•	ease   seal = seal =   seal =
	Bebbug "Rank in was reached, a solution was found " end
	explore_right_side[1] = false
	k = 1.1 while explore_right_side[k] == true
	explare_right_side( k  = false
	ond ond ond ond in kil i kil
	n_prone ++ 1
end 1 += 1	
end if 1 == 3	
end	Accord.
display(first_c return nsol, st	pocar) tarage_mol_Counter(roop_node_nop_upath_nop_ddf, r_branch_nc_prune)
end	
function classicBPs	see(2) (MRSd to 1: 1MRT type), 154,
virtual_c allmol ::	:: Float6, 8ool)
	Time is solver is not prepared to find all solutions yet*)
function classicBPs  c::Float  virtual_c  allmol::  if allmol == tx	tisk, :: Flastid,  Bool) The is solver is not prepared to find all solutions yet')

276 277	mol = MoleculaType(Wettor(AtomType)(underf.n), 0.0) for i=1:n
278 279	mol.atoms[i] = AtomType(0.0,0.0,0.0) end
280 281	C = zeros(4,4)  noal = 0
282 283	strage_nol = Dict[Int64,MoleculeType]() n_prone = 0
284	
286 287	$con_{i}$
288 289	
290 291	
292	sol.atos[1], y = 0.0 sol.atos[1], z = 0.8
294	#second aton
295 296	nol.stons[2] are = NW6dsts.info[1,2].dist nol.stons[2] are = NW6dsts.info[1,2].dist
297 298	sol. atoss [2], y = 0.0 sol. atoss [2], z = 0.0
299 300	# tird atom 012 = NMRdata:info[1,2].dist
301	D13 = NMMSdra.info[1,3].dist  D23 = NMMSdra.info[2,3].dist
303	014 = 0.0 024 = 0.0
305 306	034 = 0.0 c0,50 = bondangla(012,013,023)
307 308	noq.node == [3,6,1,1]  Cu,su = (8,8,8,8)
309	sol.atoss[3] element = NMSdata.info[3,:].ncwal[1].atoml sol.atoss[3] x = -012+023+03
311 312	sol. atoss[3] y = 023*48 sol. atoss[3] z = 0.0
313 314	cop_node == [1,2,0,0]  C = zeros(4,4)
315 316	C[1,4] = sol.atoss[3], x C[2,4] = sol.atoss[3], y
317 318	C[3,4] = nol.atons(3), z C[1,1] = c8
319 320	C[1,2] = 58 C[2,1] = 58
321 322	C(2,2) = -48 C(3,3) = -1.0
323 324	C[4,4] = 1.0 1 = 4 # branching starts at atom 4
325 326	pos: = 4 # position in virtual path explore_right_ride = zeros(Sool_n)
327 328	C_list = Array(Array(Float64,2))(undef,n) # to access level l it is need to put l-3 println(C)
329 330	C_list[3] = C C_before = zeros(4,4)
331 332	<pre>= Array(Array(Plantsi,2))(under,n) = to access level 1 it is need to put 1-3 EUG = zeros(4.4)</pre>
333 334	while less  por = finfall(NMdata.virtual.path.sel-1)[1] =1
335 336	display(1) copyto((Chefran,Clist(1-1))
337 338	keep = true while keep
339 340	try  D14 = NMGdata.info[NMGdata.virtual_path[pos-3],NMGdata.virtual_path[pos]].dist
341 342	catch  D14 = sqrt((mol. atoms[MMGata.virtual_path[pos-3]].x - mol. atoms[MMGata.virtual_path[pos]].x)*2 + (mol. atoms[MMGata.virtual_path[pos-3]].y - mol. atoms[MMGata.virtual_path[pos]].y)*2* (mol. atoms[MMGata.virtual_path[pos-3]].z - mol. atoms[MMGata.virtual_path[pos-3]].y - mol. atoms[MMGata.virtual_path[pos-3]].y - mol. atoms[MMGata.virtual_path[pos-3]].z - mol. atoms[MMGata.virtual_path[pos-3]].z - mol. atoms[MMGata.virtual_path[pos-3]].y - mol. atoms[MMGata.virtual_path[pos-3]].y - mol. atoms[MMGata.virtual_path[pos-3]].z - mol. atoms[MMGata.virtual_path[pos-3]].z - mol. atoms[MMGata.virtual_path[pos-3]].z - mol. atoms[MMGata.virtual_path[pos-3]].y - mol. atoms[MMGata.virtual_path[pos-3]].z - mol
343 344	end try
345 346	D24 = MMGdsta.info[WMGdsta.virtual_path[pos-2],WMGdsta.virtual_path[pos]].dist catch
347 348	D24 = sqrt((mol.atoms(WMSdata.virtual_path[pos-2]) x - mol.atoms(WMSdata.virtual_path[pos-2]) x - mol.atoms(WMSdata.virtual_path[pos-2]) y - (mol.atoms(WMSdata.virtual_path[pos-2]) y - mol.atoms(WMSdata.virtual_path[pos-2]) y - (mol.atoms(WMSdata.virtual_path[pos-2]) x - mol.atoms(WMSdata.virtual_path[pos-2]) x - mol.atoms(W
349	ty
350	D34 = NMRdata.info/NMRdata.virtual path/pos-11.NMRdata.virtual path/pos11.dist
351	D34 = NMEdata_info[NMEdata_virtual_gath[pos-1],NMEdata_virtual_gath[pos-1],MEGata_virtual_gath[pos-1],MEGata_virtual_cath[pos-1],v= nol_atons(NMEdata_virtual_cath[pos-1]),v= nol_atons(NMEdata_virtua
351 352 353	cath  BM = sptt((mol.atoms(WMMdata.virtual_path[pos-1]) x - mol.atoms(WMMdata.virtual_path[pos-1]) x/2 + (mol.atoms(WMMdata.virtual_path[pos-1]) y - mol.atoms(WMMdata.virtual_path[pos-1]) y/2 + (mol.atoms(WMMdata.virtual_path[pos-1]) x - mol.atoms(WMMdata.virtual_path[pos-1]) x - mo
351 352 353 354 355	cath  104 sqrt((mol.atoms(DMMGdta.virtual_path[pos-1]), x - mol.atoms(DMMGdta.virtual_path[pos-1]), y - (mol.atoms(DMMGdta.virtual_path[pos-1]), y - mol.atoms(DMMGdta.virtual_path[pos-1]), y - mol.atoms(DMMGdta.virtual_path[pos-1]), z - mol.a
351 352 353 354 355 356 357	cath  Dis sprt((mol.atoms(DMMGdta.virtual_path[pos-1]), x - mol.atoms(DMMGdta.virtual_path[pos-1]), y - (mol.atoms(DMMGdta.virtual_path[pos-1]), y - mol.atoms(DMMGdta.virtual_path[pos-1]), y - (mol.atoms(DMMGdta.virtual_path[pos-1]), y - (mol.ato
351 352 353 354 355 356 356 357 358 359	cath  10 * sqrt((mol.stoss()MMGdta.virtual_path[pos-1]].x - mol.stoss()MMGdta.virtual_path[pos-1]].y - (mol.stoss()MMGdta.virtual_path[pos-1]].y - mol.stoss()MMGdta.virtual_path[pos-1]].y - (mol.stoss()MMGdta.virtual_path[pos-1]].y - (mol.sto
351 352 353 354 355 356 357 358 359 360 361	catch  Dis sprt((mol.atoms()WMGdta.virtual_path(pos-1]), x - mol.atoms()WMGdta.virtual_path(pos-1]), y - (mol.atoms()WMGdta.virtual_path(pos-1]), y - mol.atoms()WMGdta.virtual_path(pos-1]), y - (mol.atoms()WMGdta.virtual_path(pos-1]), y - mol.atoms()WMGdta.virtual_path(pos-1]), y - (mol.atoms()WMGdta.virtual_path(pos-1]), y - (mol.atoms()WMGdta.virtual_path()mol.atoms()WMGdta.virtual_path()mol.atoms() - (mol.atoms()WMGdta.virtual_path()mol.atoms() - (mol.atoms()WMGdta.virtual_path()mol.atoms() - (mol.atoms()WMGdta.virtual_path()mol.atoms() - (mol.atoms()WMGdta.virtual_path()mol.atoms() - (mol.atoms(
351 352 353 354 355 356 357 338 359 360 361 362 363	cath  Dis sprt((mol.atoss)@Modata.virtual_path[pos-1]] x - mol.atoss[@Modata.virtual_path[pos-1]] y - mol.atoss[@Modata.virtual_path[pos-1]] y - mol.atoss[@Modata.virtual_path[pos-1]] y - mol.atoss[@Modata.virtual_path[pos-1]] x - mol.atoss
351 352 353 354 355 356 357 358 359 360 361 362 363 364 364	cath  Dis * spt:((mol.atoss;)MM6sta.virtual_path[pos-1]; x - mol.atoss;)MM6sta.virtual_path[pos-1]; x - mol.atoss;)MM6sta.virtual_path[pos-2]; MM6sta.virtual_path[pos-2]; x - mol.atoss;)MM6sta.virtual_path[pos-2]; x - mol.atoss;)MM6sta.virt
351 352 353 354 354 355 355 356 357 358 359 360 361 361 362 363 365 366 366 366 366 367	cath  Dis * sprt((mol. atoms;)MMGsta.virtual_path[pos-1]; x - smol. atoms;)MMGsta.virtual_path[pos-1]; y - smol. atoms;)MMGsta.virtual_path[pos-1]; y - smol. atoms;)MMGsta.virtual_path[pos-1]; x - smol
351 352 353 354 354 355 356 356 357 389 360 361 362 363 363 364 365 366 366 366 367 368	Catch  Dis * sprt((mol. atoms;)MMGsta.virtual_path[pos-1]; x - mol. atoms; [MMGsta.virtual_path[pos-1]; y - mol. atoms; [MMGsta.virtual_path[pos-1]; y - mol. atoms; [MMGsta.virtual_path[pos-1]; x - mol. atoms; [MMGsta.virtual_path[pos-1]; x - mol. atoms; [MMGsta.virtual_path[pos-1]; y - mol. atoms; [MMGsta.virtual_path[pos-1]; y - mol. atoms; [MMGsta.virtual_path[pos-1]]; x - mol. atoms;
351 352 353 354 355 356 357 358 360 361 362 363 364 365 363 364 365 363 364 365 367 369 370 371	cath  Dis * sprt((mol.atos) (MMSdata.virtual_path[pos-1]) x - mol.atos) (MMSdata.virtual_path[pos-2]) x - mol.
351 352 353 354 355 356 357 388 399 360 361 362 363 364 365 365 366 367 368 369 377 372 372	cath  Six sprt((mol.atos)(MMSdata.virtual_path[pos-1]) x - mol.atos)(MMSdata.virtual_path[pos-1]) y - (mol.atos)(MMSdata.virtual_path[pos-1]) x - mol.atos)(MMSdata.virtual_path[pos-1]) x - mol.atos
351 352 353 354 355 356 357 388 360 361 362 363 364 365 366 367 368 369 377 372 372 373 374 374	cath  Dis * sprt((mol.atos) (MMSdata.virtual_path)pos-1]; x - mol.atos (MMSdata.virtual_path)pos-1]; x - mol.at
351 352 353 354 355 356 357 388 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 377	State   Stat
351 352 353 354 355 356 357 388 399 300 361 362 363 364 365 367 368 369 377 378 379 371 372 373 375 377 378 379 370 377	Set of the part (mol. stons [MMSdata virtual_path[pos-1]] x - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] y) - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] y) - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] x - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] x - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] x - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] x - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] x - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] x - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] x - mol. stons [MMSdata virtual_path[pos-1]] y - mol. stons [MMSdata virtual_path[pos-1]] x - mol. stons [MMSdata virtual_path[pos-1]] y - mo
351 352 353 354 355 356 357 358 360 361 362 364 365 366 367 368 369 370 371 372 372 373 374 375 376 377 378 379 379 379 379 379 379 379 379	Catch
351 352 353 354 355 356 357 358 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 378 379 379 370 371 372 373 374 375 377 378 379 379 370 370 371 372 373 374 375 377 378 379 370 370 370 370 370 370 370 370	Catch
351 352 353 354 355 355 356 357 360 361 362 363 364 364 365 366 367 377 378 379 371 372 373 374 375 377 378 379 379 370 371 372 373 374 375 376 377 378 379 379 370 370 371 372 373 374 375 376 377 377 378 379 379 370 370 370 370 370 370 370 370	Cath   City   Cath
351 352 353 354 355 355 356 357 358 360 361 362 363 364 364 365 366 367 370 371 372 373 373 374 375 377 377 378 379 379 379 379 379 379 379 379 379 379	Discription   Interpolate (Prince   Interp
351 352 353 354 355 355 356 357 358 360 361 362 363 364 362 363 364 364 365 366 367 370 371 372 373 373 374 375 377 377 378 379 379 379 379 379 379 379 379 379 379	Carto   State   Septic (red. inters) (Whites virtual_path) (post   1) a * and a state   Whites
351 352 353 354 355 356 357 358 359 360 361 362 362 363 366 367 370 371 373 373 374 375 379 379 379 379 379 379 379 379 379 379	Carcia
351 352 353 354 355 356 356 357 358 360 361 362 363 365 366 367 370 371 372 373 374 375 377 377 378 379 380 381 381 382 383 383 383 384 385 386 387 388 388 389 399	Company   Comp
351 352 353 354 355 356 357 358 360 361 362 363 364 362 366 366 367 377 373 373 374 375 377 378 379 380 381 381 382 383 384 385 388 389 390 390 391 392	Card   Str.   Sept.   Card   Str.   Sept.   Card   Str.   Sept.   Se
351 352 353 354 355 356 356 357 358 360 361 362 363 364 362 366 366 377 378 372 373 373 374 375 379 380 381 381 382 383 384 385 386 387 389 380 381 382 383 384 385 389 380 390 390	Carlo
351 352 353 354 355 356 356 357 358 360 361 362 363 364 364 362 366 366 377 378 377 377 377 377 378 378 379 388 381 381 382 383 384 385 387 389 389 390	Cold
351 352 353 354 355 355 356 357 358 360 361 362 363 366 367 363 366 367 377 371 372 373 378 378 379 378 379 378 379 379 370 371 371 372 373 373 374 375 377 378 379 379 370 370 371 371 372 373 374 375 377 378 379 379 370 370 371 371 372 373 374 375 377 378 379 379 370 370 370 371 371 372 373 373 374 375 377 378 379 379 370 370 370 370 371 371 372 373 374 375 377 378 379 379 370 370 370 370 370 370 370 370 370 370	Cold   St. top((cold.sep)   (st. est.   state)
351 352 353 354 355 355 356 357 358 360 361 362 363 366 367 377 373 372 373 374 375 379 370 371 371 372 373 373 374 375 379 379 370 371 371 372 373 373 374 375 379 370 371 371 372 373 373 374 375 379 370 370 371 371 372 373 372 373 374 375 379 370 370 371 371 372 373 372 373 374 375 379 370 370 370 371 371 372 373 373 374 375 379 370 370 370 370 370 371 371 372 373 373 374 375 379 370 370 370 370 370 370 370 370 370 370	Cold
351 352 353 354 355 355 356 357 358 360 361 362 363 366 369 367 366 369 377 378 379 378 379 379 379 379 379 379 379 379 379 379	The content of the spring in the content product of the content products are all products
351 352 353 354 355 355 356 357 358 360 361 362 362 363 364 362 366 367 370 371 373 373 374 375 378 379 378 379 379 379 379 379 379 379 379 379 381 381 387 388 386 387 388 386 387 388 387 388 387 388 389 390 391 392 393 394 401 405	Company
351 352 353 354 355 356 357 358 359 360 361 362 363 364 362 363 366 367 376 377 378 378 379 380 381 381 381 381 382 383 386 387 387 388 388 388 388 388 389 390 391 392 393 394 401 405 402 405	The satisfied interplaces strong proposed to stro
351 352 353 354 355 356 356 357 358 360 361 362 363 364 362 366 370 371 373 373 375 378 379 380 381 381 381 382 383 384 385 386 387 388 388 388 389 389 390 390 390 390 390 390 390 390 390 39	The Company of the
351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 366 370 371 372 373 373 373 373 374 375 379 380 381 381 382 383 381 381 382 383 383 381 383 383 383 383 384 385 387 388 389 389 389 389 399 400 400 401 402 402 406 407 407 408 406 407 407 408 406 407 408 406 407 408 406 407 408 406 407 408 406 407 408 406 407 408 406 407 408 406 407 408 406 407 408 408	Company
351 352 353 354 355 356 356 357 358 359 360 361 362 363 364 364 365 366 370 371 372 372 376 377 372 378 379 373 379 374 377 372 378 379 379 379 379 379 379 379 379 379 379	The Company of the

417 418		end end
419 420		
421 422		if explore_right_side[1] == true  8[1] = torsionmatrix(B[1])
423 424		#nop_node == [0,0,0,0]  Clist[] = prodestrix(Clist[-1],8[1])# tenho que otimizar este calculo
425 426		rsp_rode == [24,33,8,8] end
427 428		mol.stons(1).x = C_list(1)[1,4]
429		mol.atons([1], y = C_list([1], 4] mol.atons([1], z = C_list([1], 3,4]
430 431		$count = [0,0,0,0]$ $\rho$ , $count = pransingtest(mol,1,1)MHodata, \epsilon, count) #preciso modificar$
432 433		<pre>comp_ddf += count  #println("C = Cbefore*B at level 5[1] right side   S[C_List[1]]) = S[C_List[1-1]] * S[B[1]]")</pre>
434 435		if p = 1 of ton
436 437		# println("Partial solution by right side at level \$(1) ", mol)  n_branch = 1
438		1 += 1 else
440 441		
442		@debug "Rank n was reached, a solution was found "
444		end else
445 446		if explore_right_side[1] == true explore_right_side[1] = false
447 448		<pre>k = 1-1 while explore_right_side[k] == true</pre>
449 450		esplore_riph_tide(s) = false k = 1
451 452		end explore_right_ride[k] = true
453 454		1 = K 0,gune == 1
455 456		eplore_right_ride[]] = true
457		expose_tignc_lose[i] = time  end
458 459		end
460 461		return nsol, storage_mol,Counter(nop_mode,nop_wpath,nop_deff,n_branch,n_prune) end
462 463		
464 465		
466 467		**-   **-
468 469		classics: : Function
470 471		This function is an important solver of this package. This solver implements the algorithm given in
471 472 473		Liberti, L., Lavor, C., & Maculam, M. (2008). A branch-and-prove algorithm for the solecular distance geometry problem. International Transactions in Operational Research, 15(1), 1-17.
474		The main difference of our implementation is that the input data ('NMRType') can store informations of preprocessing function and as consequence we can optimize the search tree of this algorithm.
475 476		function classis@PUMMdata:: MMType,
477 478		c :: Float64,  wirtual_c :: Float64,
479 480		alkei : sool, tre_limit)
481 482		#start = Dutes.now() #time_elapsed = Second(0.0)
483		14600 (F).HB) immyles port in classics?  1 (188-000) (m.) 1400 (m.
484		n = NMSdxta.din
485 486		if n < 3  ArgumentError('Invalid dimension of NMRdsta')
487 488		and .
489		virtual_r' = virtual_r*virtual_r
490 491		nsol = 0 storage_mol = Dict(Int64,MoleculeType)()
492		1 (0.000ME) samples spent in initialization 0 (cx.), 1 (100.00%) (incl.) when called from classic@P linn.640
493		function initialization()
495		nol = NoisculeType(Vector(AtonType)(undef,n),0.0)  for i=1:n
496 497		nol.atoms[i] = AtomType(0.8,0.8,0.8)  end
498 499		C > 2000(4.4)
500 501		# first atom nod atoms(1), elemen = NMEdata.info[1, 2) naval[1] atoms
502 503		nol.atons(1), $x = 0.0$ nol.atons(1), $y = 0.0$
504		nol.atons(I), $x = 0.0$ Faccord aton
506	1 (0.0040%	section and 1 1880.00% spales spont calling <u>satindes</u> nol.atoms[2].element = NWGdats.info[2,:].nvval[1].atoml
507		mol.atoms[2].x = -NMRdata.info[1,2].dist
509		nol. atoms(2], $y = 0.0$ nol. atoms(2], $z = 0.0$
510 511		# tird atom  Ol2 = NMRAdra.info[1,2].dist
512 513		013 = NMRdata.info[1,3].dist  023 = NMRdata.info[2,3].dist
514 515		6,36 * bondergl(01,03,032)  nol.atens(13); leaens * wideta.infe(1,1); noval(1):atent
516 517		nol.atons[3].x = -0124-023+08 nol.atons[3].y = 023+69
518 519		mol. atom(3); $z = 0.8$ C = 2eros(4,4)
520 521		C[1,4] = sol. atos[3], x  C[2,4] = sol. atos[3], y
522		C(3,4) = nol.atons(3).z
523 524		C(1,1) = c8 C(1,2) = c8
525 526		C[2,1] = s8 C[2,2] = -c8
527 528		C(3,3) = -1.0 C(4,4) = 1.8
529 530		return 4,4,m01,C
531 532		end .
533 534		# defining closure  function classis®_closure[1:: Int64,
535		populitifs, sol:: Maleculatys,
536		nol :: MoleculaType,  C :: Array(Float64,2))
		#time_elapond = Dotes.now()-start
538 539		### ### ##############################
538 539 540 541		#end
538 539 540		14648 (57.87%) samples spent in classic8P_closure
538 539 540 541		648 (98.78%) (ex.), 14639 (99.88%) (incl.) when called from classicsP_closure line 512 (528 (49.72%) (ex.), 14639 (99.88%) (incl.) when called from classicsP_closure line 513 (538 (49.72%) (ex.), 14639 (99.88%) (incl.) when called from classicsP_closure line 513 (538 (538 (548 (548 (548 (548 (548 (548 (548 (54
538 539 540 541 542 543	5 (0.06%) 843 (3 33%)	641 (59.7%) (ex.), 1463 (79.8%) (incl.) when called from classic#F_closure time 517 (20.7%) (ex.), 2463 (79.8%) (incl.) when called from classic#F_closure time 517 (20.7%) (ex.), 1463 (79.8%) (incl.) when called from classic#F_closure time 518 (20.7%) (ex.), 1464 (108.8%) (incl.) when called from classic#F_closure time 518 (20.7%) (ex.), 1464 (108.8%) (incl.) when called from classic#F_closure time 518 (20.7%) (ex.), 1464 (108.8%) (incl.) when called from classic#F_closure time 518 (20.7%) (ex.), 1464 (108.8%) (incl.) when called from classic#F_closure time 518 (20.7%) (incl.) (incl.
538 539 540 541 542 543	5 (0.06%) 843 (3.33%	541 (97.7%) (ex.), 1489) (99.8%) (incl.) when called from classic#F_closure <a href="https://doi.org/10.1%">https://doi.org/10.1%</a> (98.8%) (incl.) when called from classic#F_closure <a href="https://doi.org/10.1%">https://doi.org/10.1%</a> (98.), 14648 (100.0%) (incl.) when called from classic#F_closure <a href="https://doi.org/10.1%">https://doi.org/10.1%</a> (98.), 14648 (100.0%) (incl.) when called from classic#F_closure <a href="https://doi.org/10.1%">https://doi.org/10.1%</a> (98.)

47		1 (0.0040%)	1 (180.80%) samples spent calling <u>gatindax</u>
48		,	virtualisatives = NMMata.virtual_path[pos-1]  5 (71.43%) samples spont calling outlander
	0.0040%)	7 (0.03%)	1 (4.79) samples sport calling =:  1 (Metalax virtual_path(pox-1) == virtualPos  1 (Metalax virtual_path(pox-1) == virtualPos
50 51			514 * 6.8 else
52		193 (0.76%)	43 (22 28%) samples spent calling gatindes 99 (51.38%) samples spent calling gatindes (1.62-47%) samples spent cal
53			014 = NMRdata.info[NMRdata.virtual_path[por-3],virtualFos].dist end
	).0040%)	1 (0.0040%)	if MMRdsta.virtual_path[pos-2] == virtualPos
55 56			024 = 0.0 else
57		47 (0.19%)	6 (12.7%) samples sport calling <u>extinder</u> 19 (40.43%) samples sport calling <u>extinders</u> 2 (46.31%) samples sport calling <u>extinders</u> 3 (46.31%) samples sport calling <u>extinders</u>
58			024 = NMRdsta.info[NMRdsta.virtual_path[pos-2],virtualFos].dist end
59 60			if virtualisst#es == virtual#es
61			034 = 0,0 else
63		33 (0.13%)	17 (S1.5XX) samples spent calling <u>entiodex</u> 5 (IS.5XD) samples spent calling <u>entiodex</u> 11 (33.3XD) samples spent calling <u>entioneexty</u> 11 (33.3XD) samples spent calling <u>entioneexty</u>
64		33 (0.13%)	D34 = NMMGdxta.info[virtuallastFos.virtualPos].dist
65			end 3 (100.00%) samples spent calling <u>optindes</u>
66 67		3 (0.01%)	1f WMGdta.virtual_gath[pos-2] == NMddta.virtual_path[pos-2] D12 = 0.0
68			else 9 (10.11%) samples sport calling <u>optimizer</u>
69		89 (0.35%)	48 (44.34) samples spent calling <u>settioners</u> 48 (44.34) samples spent calling <u>settioners</u> 48 (44.34) samples spent calling <u>settioners</u> 49 (44.34) samples spent calling <u>settioners</u> 40 (44.34) samples spent calling <u>settioners</u> 41 (44.34) samples spent calling <u>settioners</u> 42 (44.34) samples spent calling <u>settioners</u> 43 (44.34) samples spent calling <u>settioners</u> 44 (44.34) samples spent calling <u>settioners</u> 45 (44.34) samples spent calling <u>settioners</u> 46 (44.34) samples spent calling <u>settioners</u> 46 (44.34) samples spent calling <u>settioners</u> 47 (44.34) samples spent calling <u>settioners</u> 48 (44.34) samples spent calling settioners 48 (44.3
70			012 = NMRdsts.info(NMRdsts.virtual_path[pos-3], NMRdsts.virtual_path[pos-2]].dist end
71		3 (0.01%)	2 (6.5 TM) samples spent calling <u>entineers</u> 1 (33.33%) samples spent calling <u>entineers</u> 1 (MMEdian virtual_apthipes-3) = wirtuallastFos
72			0 3 = 0'0   0 3
74		42 (0.17%)	25 (93.73) amples spent calling outlining. 14 (31.33) amples spent calling outlining.
75		- (3.1779)	3 (7.14%) camples spent calling getinodes  D13 = WMMGata.info[WMGata.virtual_path[pos-3],virtuallastFos].dist  end
٠.		6 (0.02%)	end 1 (16.57%) samples spent calling <u>entroperty</u> 4 (66.57%) samples spent calling <u>entroperty</u>
77	0.0040%)		<pre>if (MM6data.virtual_path(pos-12) == virtualLastFos</pre>
78			else 412 de 12 may samples spent calling <u>sminning</u>
79		31 (0.12%)	13 (41.54%) samples spent calling <u>editoriests</u> 14 (45.16%) samples spent calling <u>editories</u> 103 = WMEdata.infolWMEdata.virtual.path[pos-2], virtualiastFoo].dist
80 81			end end
			\$ (23.81%) camples spont calling bondwords 1 (4.78%) camples spont calling bondwords 2 (9.52%) camples spont calling bondwords 2 (9.52%) camples spont calling bondwords 3 (9.52%) camples spont calling bondwords 4 (9.52%) camples spont calling bondwords 4 (9.52%) camples spont calling bondwords 5 (9.52%) camples spont calli
82		21 (0.08%)	6 (28.57%) ramples sport calling bondample 5 (28.8%) ramples sport calling bondample
4			2 (9.52%) samples spent calling handmodel  dip.de = hondmodel(023,004,004)  1.6.83%) respirator post calling handmodel  dip.de = hondmodel(023,004,004)
83		20 (0.08%)	1 1.5 ANN; camples sport calling <u>landrasionnessels</u> 10 (500 NN) samples sport calling <u>landrasionnessels</u> 11 (5.00) samples sport calling <u>landrasionnessels</u> 11 (5.00) samples profer calling <u>landrasionnessels</u>
_			8 (48,00%) samples spent calling <u>badforstionsole</u> cu <sub>1</sub> , su <sub>2</sub> = badforstionningle(012,033,04,034)  112 (179,40%) spendes spent calling <u>forstionnistris</u>
84 (1 85	1.12%)	1404 (5.55%)	8 = torsionmatrix(ch,s0,cu,su,034)  if levirtualPos
	(0.02%)	1042 (4.12%)	1898 (99.62%) samples sport calling <u>productis</u> C = productis(C_before, B)
87 88			break else
89		51 (0.20%)	33 (64.7%) camples spent calling optionerry 13 (13.7%) camples spent calling optionerry 14 (13.7%) camples spent calling optionerry 15 (13.7%) camples spent calling optionerry
90			cpx = nol.atons[virtualFos].x $cpy = nol.atons[virtualFos].y$
91		2 (0.0079%)	2 (100.00%) samples spent calling <u>optionosetty</u> cpr = mol.atoms[virtualPor].2
		040 (0 500)	1 (9.1%) samples spent calling graduatrix 915 (99.5%) samples spent calling graduatrix
	0.0079%)	919 (3.63%)	1 (8.1%) samples spent calling <u>anomatrix</u> Virtual_Torsion = prodmatrix(C_before,8)
94			2 (25.00%) samples sport calling 2 1 (12.50%) samples sport calling options
95		8 (0.03%)	2 (25.80%) samples sport calling 2 2 (25.80%) samples sport calling 1 1 (12.80%) samples sport calling 1 1 (12.80%) samples sport calling 1
2			if (Wirtual_Torsion[1,4]- cps)^2=(Wirtual_Torsion[2,4]- cpy)^2=(Wirtual_Torsion[3,4]- cpz)^2> wirtual_t^2  57 (96.61%) samples spent calling <u>Torsionnatris</u>
	7.007570)	59 (0.23%)	48 (180.00%) samples spent calling graduatis
97 98		48 (0.19%)	
99			C_before = productrix(C_before, B) else
00		108 (0.43%)	C_before = productrix(C_before,8) else else 108 (108.000) implies sport calling (200)
01		108 (0.43%)	C_before = prodmatrix(C_before, B) else
01 02 03		108 (0.43%)	Clefters = productizi(Cleftus,8) else else 100 (100 AMD; Implies spent calling GGM; Clefters = copy(Virtual_Torsion) end
01 02		108 (0.43%)	Clefror = prodestrix(Defoxe,8) else else 108 (100.00%) samples sport calling GGG( Clefror = copy(Virtual_Tersion) end pos = pos+1 end
01 02 03 04		108 (0.43%) 6411 (25.33%)	C_before = productiva(_before,8)  alse  alse  106 (108.0MS) samples spent calling GGG
)1 )2 )3 )4		6411 (25.33%)	Clefers = prodestrix(Clefers.8)
01 02 03 04			Clefters = productive(_before, 8)  alse  alse  lime (imm. dem) suspice spent calling GGG  Clefters = copy(virtual_Tension)  end  pos = pos*1  end  end  end  6272 (97.RRM) sumples spent calling setiones  81 (1.790, samples spent calling setiones  13 (8.970, samples spent calling setiones  31 (8.970, samples spent calling setiones  31 (8.970, samples spent calling setiones  31 (8.970, samples spent calling setiones  42 (75.890, samples spent calling setiones  43 (75.890, samples spent calling setiones  44 (75.890, samples spent calling setiones  45 (75.890, samples spent calling setiones  46 (18.791) samples spent calling setiones  47 (75.890, samples spent calling setiones  48 (75.890, samples spent calling setiones  49 (75.890, samples spent calling settiones  40 (75.990, samples spent calling settiones  41 (75.990, samples spent calling settiones  42 (75.990, samples spent calling settiones  43 (75.990, samples spent calling settiones  44 (75.990, samples spent calling settiones  45 (75.990, samples spent calling settiones  46 (18.791) samples spent calling settiones  47 (75.990, samples spent calling settiones  48 (75.990, samples spent calling settiones  49 (75.990, samples spent calling settiones  40 (75.990, samples spent calling settiones  41 (75.990, samples spent calling settiones  42 (75.990, samples spent calling settiones  43 (75.990, samples spent calling settiones)
01 02 03 04 05 06		6411 (25.33%)	Clefters = productive(_before, B)  alse  alse  188 (188.08E) samples sport calling GGE)  Clefters = copy(Virtual_Tension)  end  pos = pos-1  end  end  end  227 (97.2EM) samples sport calling setiodes  90 (1.3PS) samples sport calling setiodes  90 (1.3PS) samples sport calling setiodes  18 (8.9PS) samples sport calling setiodes  18 (8.12.2PS) samples sport calling setiodes  18 (8.12.2PS) samples sport calling setiodes  18 (18.2PS) samples sport calling setiodes
11 12 13 14 14 15		6411 (25.33%) 56 (0.22%)	Claffors = productive(Claffors.8)  alse  alse  188 (198.08%) swaples spent calling cost  Claffors = copy(Virtual_Torsion)  end  pos = post1  end  end  227 (07.28%) swaples spent calling estimates  (pol.1990) swaples spent calling estimates  12 (0.79.28%) swaples spent calling estimates  42 (75.28%) swaples spent calling estimates  42 (75.28%) swaples spent alling estimates  43 (75.28%) swaples spent calling estimates  44 (75.28%) swaples spent calling estimates  45 (75.28%) swaples spent calling estimates  46 (10.79%) swaples spent calling estimates  47 (75.28%) swaples spent calling estimates  48 (75.28%) swaples spent calling estimates  49 (10.79%) swaples spent calling estimates  40 (10.79%) swaples spent calling estimates  41 (10.79%) swaples spent calling estimates  42 (75.28%) swaples spent calling estimates  43 (10.80%) swaples spent calling estimates  44 (10.79%) swaples spent calling estimates  45 (10.79%) swaples spent calling estimates  46 (10.79%) swaples spent calling estimates
001 002 003 004 005 006		6411 (25.33%) 56 (0.22%) 3 (0.01%)	Clefters = productive(_before, B)  alse  alse  188 (188.08E) samples sport calling GGE)  Clefters = copy(Virtual_Textion)  end  pos = pos*1  end  end  end  227 (97.2EM) samples sport calling setiodes  90 (1.3PS) samples sport calling setiodes  90 (1.3PS) samples sport calling setiodes  188 (8.9PS) samples sport calling setiodes  80 (1.3PS) samples sport calling setiodes
011 022 033 044 055 066		6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%)	Clasters = productiva(Clasters, B)  alse  alse  188 (198.080) samples spent calling SSSS  Clasters = copy(virtual_Torsion)  end  pos = post1  end  end  227 (97.283) samples spent calling setions  (87.2 (97.283) samples spent calling setions  19 (1.393) samples spent calling setions  19 (1.993) samples spent calling setions  19 (1.993) samples spent calling setions  19 (1.993) samples spent calling setions  10 (1.993) samples spent calling setions  20 (1.993) samples spent calling setions  21 (1.98 SSS) samples spent calling setions  22 (1.98 SSS) samples spent calling setions  23 (1.98 SSS) samples spent calling setions  24 (1.98 SSS) samples spent calling setions  25 (1.98 SSS) samples spent calling setions  26 (1.87 SSS) samples spent calling setions  27 (1.98 SSS) samples spent calling setions  28 (1.98 SSS) samples spent calling setions  29 (1.99 SSSS) samples spent calling setions  20 (1.98 SSSS) samples spent calling setions  20 (1.98 SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
11 12 13 13 14 15 16		6411 (25.33%) 56 (0.22%) 3 (0.01%)	Cleftors = productivat(_before,B)  alse  alse  188 [180.08] semples spent calling gags  Cleftors = copy(Virtual_Texton)  and  get  get  get  od  od  od  202 [77.28] semples spent calling settomer  302 [77.28] semples spent calling settomer  20 [78.78] semples spent calling settomer  30 [79.78] semples spent calling settomer  40 [79.78] semples spent calling settomer  50 [79.78] semples spent calling settomer  61 [79.78] semples spent calling settomer  81 [79.78] semples spent calling settomer  82 [79.78] semples spent calling settomer  83 [79.78] semples spent calling settomer  84 [79.78] semples spent calling settomer  85 [79.78] semples spent calling settomer  85 [79.78] semples spent calling settomer  86 [79.78] semples spent calling settomer  87 [79.78] semples spent calling settomer  88 [79.78] semples spent calling settomer  89 [79.78] semples spent calling settomer  80 [79.78] semples spent calling settomer  80 [79.78] semples spent calling settomer
011	08	6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%)	Claffore = productivat(_before,B)
001	08 3.43%)	6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%)	Clefters = productiva(_before,B)  alse  alse  188 [180.08] suspice spent calling uses  Clefters = copy(Virtual_Textion)  end
001   002   003   004   005   005   006   007   007   008   009	08 0.43%)	6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%) 1187 (4.69%) 2717383	Clare   productis(Clarifore, B)
001   002   003   004   005   005   006   007   007   008   009	08 0.43%)	6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%) 1187 (4.69%) 4 (0.02%)	Cleffors sproductis(_leffors.B)
001   002   003   004   005   006   007   008   009   007   008   009   007	08 0.43%) 0.0079%) 38 0.94%)	6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%) 1187 (4.69%) 4 (0.02%) 2717383 (10735.55%)	Claritors productiva(_laritors_B)  sits  sits  sits  sits (188.080) imples speet calling gaze  and  par *port  and  par *port  and  par *port  and  sits (1.380) imples speet calling statutes  par *port  and  sits (1.380) imples speet calling statutes  sits (1.380) imples speet calling
001   002   003   004   005   006   006   007   007   008   009   007	08 0.43%) 0.0079%) 38 0.94%)	6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%) 1187 (4.69%) 2717383	Carton s productive(_lefters.B)  size  size  120 (180.000) camples speet calling oggs  per sport  end  per sport  end  627 (97.883) camples speet calling oggstides  827 (97.883) camples speet calling oggstides  828 (150.000) per sport  829 (150.000) per sport oggstides  821 (879) camples speet calling oggstides  82 (879) camples speet calling oggstides  83 (879) camples speet calling oggstides  84 (879) camples speet calling oggstides  85 (879) camples speet calling oggstides  86 (879) camples speet calling oggstides  87 (879) camples speet calling oggstides  88 (879) camples speet calling oggstides  89 (879) camples speet calling oggstides  90 (879) camples speet calling oggstides  11 unples oper calling oggstides  12 unples oper calling oggstides  13 unples oper calling oggstides  14 unples oper calling oggstides  15 unples oper calling oggstides  16 unples oper calling oggstides  17 unples oper calling oggstides  17 unples oper calling oggstides  18 unples oper cal
001 002 003 004 005 005 006 006 007 007 008 009 009 009 009 009 009 009 009 009	08 0.43%) 0.0079%) 38 9.94%)	6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%) 1187 (4.69%) 4 (0.02%) 2717383 (10735.55%)	Clarifies productive(Carlins,D)  the  the  the  the  the  the  continued of the continued o
001	08 0.43%) 0.0079%) 33 38 0.0040%)	6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%) 1187 (4.69%) 2717383 (10735.55%) 1 (0.0040%) 4 (0.02%)	Charton = specialization (Labelana s)  All 188 (1890 temples spect calling ages  Charton = conjugation (Labelana s)  ord  Off  Off  Off  Off  Off  Off  Off  O
001 001 002 003 003 004 005 006 006 006 007 007 008 008 009 009 009 009 009 009 009 009	08 0.43%) 0.0079%) 33 38 0.0040%)	6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%) 1187 (4.69%) 4 (0.02%) 2717383 (10735.55%)	Chefform a productive (Leafon and Section 2006)  100 (Leafon a computation) (Leafon and Section 2006)  100 (Leafon a computation) (Leafon and Section 2006)  100 (Leafon a computation) (Leafon and Section 2006)  100 (Leafon and Section 2006) (Leafon and Section 2006)  101 (Leafon and Section 2006) (Leafon 2006) (L
11	08 0.0079%) 0.0079%) 0.0040%) 0.0040%)	6411 (25.33%) 56 (0.22%) 3 (0.01%) 4 (0.02%) 1187 (4.69%) 2717383 (10735.55%) 1 (0.0040%) 4 (0.02%)	Charton a producting Caption, 20  188 (188 (1891) suspins sport colling 1920  Charton a copyrimal fraction  of the part of the

624	20 (0.08%)	255 (1.01%)	235 (92.18%) samples spent calling <u>gnodmatrix</u> C = prodmatrix(C_before,8)* tenho que otimizar este calculo
П			5 (12.28) samples spent calling settroperty/ 20 (51.28) samples spent calling settroperty/
625		39 (0.15%)	14 (15.90) camples sport calling <u>editorsectiv</u> mod_store(); x = (1,4)
626		6 (0.02%)	3 (50 MPC) samples sport calling settomoserty 1 (16.57%) samples sport calling settomose 2 (33.33%) samples sport calling settomose 2 (33.33%) samples sport calling settomose 3 (50.33%) samples sport calling settomose 3 (50.33%) samples sport calling settomose 4 (50.33%) samples sport calling settomose 5 (50.33%) samples 5 (50.33%) sam
		- ()	nol.atons[1], y = C[2,4]  1 (55.80) samples spont calling <u>netroporty</u>
627		4 (0.02%)	1 (2.5 ws) samples Spen calling <u>outposers</u> ) 3 (75.80%) samples spen calling <u>outposers</u> ) 80.3toss[1], z = C[3,4]
628			
	40.00.0700	445 /4 5 40/1	388 (91.490) samples spent calling grantingtes: 10 (9.480) samples spent calling grantingtes: 4 (9.980) samples spent calling grantingtes (96.100) 1 (9.280) samples spent calling grantingtes (96.100) 1 (9.280) samples spent calling grantingtes 1 (9.280) samples spent calling grantingte
629	18 (0.07%)	415 (1.64%)	3 (0.72%) samples spent calling <u>cruningtest</u>
630	2 (0.0079%)	2 (0.0079%)	if puningtes[no.],],WMdata,c) &preciso modificar  if lon
	(0.0079%)		1 samples spent calling classis@-classure 4 samples spent calling oppdatisk
631		7347994	14457 (0.28%) samples spent calling classiff closure 1 samples scent calling orthography
	(1.15%)	(29029.69%)	4 samples spent calling <u>ornalogicax</u> 5 samples spent calling <u>ornalogicax</u> 5 samples spent calling <u>ortalogica</u> 5 samples spent calling <u>ortalogica</u>
632			ClassisP_closume(1+1,pos+1,ed),C) else
633	15 (0.06%)	15 (0.06%)	nsa = nsa -1 starage_nsa [nsa ] = copy(nsa ] = copy(nsa ]
635 636			return end
637 638			end end # closure
639			1 (100 ADN) samples spent calling initialization
640		1 (0.0040%)	
641		14648 (57.87%)	dastire_dourre_L
	1 (0.0040%)	1 (0.0040%)	return real, storage_mal
643 644			end #solver classicSP
645 646			
647 648			quaternionSP :: Function
649 650			This function defines a new solver. The implementation follows ideas describing in:
651 652			Fidalgo, F. Using Quatermion Geometric Algebra for efficient rotations in Branch and Prune Algorithtm to solve the Discretizable Molecular Distance Geometry Problem. In: Proceedings of MGMSSE 2018, Campinas-SP, Brazil.
653 654			function quaternion@P(MPGata:: MMType,
655 656			c :: Float64,
657			virtual_c :: Float64,  allaol :: Bool, time_limit)
658 659			#start = Dates.now()
660			Fitns_lapsed = Second(8)  1336 (4) 233 supples sport in quaternicodP
662			0 (ex.), 18206 (180.80%) (incl.) when called from conformation lies 42  n = 1000data.dim
663 664			if n < 3  ArgumentError("Invalid dimension of NMRdata")
665 666			end .
667			virtual_t' = virtual_t'virtual_t  nool = 0
669			nous - " storage_ool = DictInt64, Noleculatype)()
670 671			function sattablization()
672 673			# TODO: (Emerson) no criação desse vetor você não pode já estabelecer um valor default para atoms? nol = MoleculeType(Vector(AtomType)(undef_n), 0.0)
674 675			for i=1:n  nol.atons[i] = AtonType(0.0,0.0,0.0.0)
676 677			end  (= Quaternian((8,8,8,8,8,8))
678 679			# first atom
680 681			mol.stoss[]], elemen = WWMsts.info[[,:],mvxl[]],stosi mol.stoss[],s = 0.8
682 683			nol.atens[1], y = 0.0 nol.atens[1], z = 0.0
684 685			#Second atom  nol.atoms[2].element = NMMdata.info[2,:].neval[1].atom]
686 687			mol.atoms[2].x = -NMRdata.info[1,2].dist
688			nol.stons[2].y = 0.0 nol.stons[2].z = 0.0
689 690			# tird atom  D12 = NMMGMata.info[1,2].dist
691 692			013 = NMRGata_info[1,3].dist 023 = NMRGata_info[2,3].dist
693 694			c8, s8 = qbondangle(012, 013, 023) Q = Quaternion(8, 8, -c8, -s8, 8, 8)
695 696			d = 2.0*023  gpol = Quaternion(0.8,d*(c0*c0-0.5),d*(c0*s0),0.0)
697 698			nol.atons[3], element = WM6drat.info[3,:].noval[1].aton1 nol.atons[3], x = qnol.v1 + nol.atons[2], x
699 700			nol.stos[3], y = qnol.v2 + nol.stos[2], y nol.stos[3], z = qnol.v3 + nol.stos[2], z
701 702			return 4.4, no.1.0
702 703 704			20cum 4.4, NO.1, Q end
705			# defining closure
706 707			function quaternion@p_closure[1 :: Int64, pos::Int64,
708 709			mal:: MaleculeType, Q:: Quaternion)
710 711			<pre>#Time_clapsed = Dates.now()-start</pre>
712 713			fif time_elapsed=time_limit i& l-n  # error("Time limit reached without found a solution!")
714			#end
715			10206 (40.20) imples sport in quaterniceSF_cloure 469 (51.50%) (xo.) 1050 (99.90) (incl.) when called from quaterniceSF_cloure line SI3 412 (x7.13%) (xc.), 1050 (99.90) (incl.) when called from quaterniceSF_cloure line SI3 (xc.), 10206 (90.900) (incl.) when called from quaterniceSF_cloure line SI3 (xc.), 10206 (80.900) (incl.) when called from quaterniceSF_cloure line SI3 (xc.), 10206 (80.900) (incl.) when called from quaterniceSF_cloureSI3 (xc.), 10206 (xc.)
	10.65	40.00	
717	(0.05%) د ،	13 (0.05%)	lastpos = 1  D34 = 0.0
718 719			virtualistF0s = 0.0 a = 0.0
720 721			b=0,0 c=0,0
722		30 t0 0685	d = 0.0 20 (100 00%) samples spent calling <u>copy</u>
723 724		20 (0.08%)	Q_before = capy(Q)
725			while true 3 (3.990) samples spent calling getindes
726		77 (0.30%)	5 (6.49%) samples speet calling statements (9) (9) (9) (5) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1
727		4 (0.02%)	virtualPos = NMMdata.virtual_gath[pos] 4 (100 ADM) samples spent calling <u>optioder</u>
727		→ (U.UZ%)	virtuallastPos = NMRdsta.virtual_path[pos-1]
729		3 (0.01%)	1 (33.33%) samples spent calling <u>set</u> 2 (66.67%) samples spent calling <u>setimoder</u>
730		· ·	if MMRGata.virtual_gath[pos-3] == virtualPos D14 = 0.0
731			else 25 (14.77); samples spent calling <u>ogtiones</u> (7 (28.14%); samples spent calling <u>ogtiones</u> (8 (28.14%); samples spent calling ogtiones (8
732		167 (0.66%)	95 (56.89%) samples spent calling <u>getinder</u>
			014 = NMRdata.info[NMRdata.virtual_path[pos-3],virtualFos].dist

13			end
84 85			if NWMSdata.virtual_path[pos-2] == virtualPos  D24 = 0.0
86		39 (0.15%)	clus 7 (17.9%) samples spent calling gatindes 19 (48.7%) samples spent calling gatindes 19 (48.7%) samples spent calling gatindes 18 (38.3%) samples spent calling gatindes 18 (38.3%) samples spent calling gatindes 19 (38.3%) samples spent calling gatindes
18			D24 = NMMdsta.info[NMMdsta.virtual_path[pos-2],virtualPos].dirt ond
10			if virtualisatPos == virtualPos
12			034 = 0.0  elice 8 (19.8%) amplie sport calling quindes
13	4	42 (0.17%)	30 (F.S. 103) sepies sport calling matindes 14 (31.33) sepies sport calling matindes 14 (31.34) sepies sport calling matindes 154 = NMMdata.info[virtualLatVex,virtualPes].dist 16 (31.34) sepies sport calling matindes 17 (31.34) sepies sport calling matindes 18 (31.34) sepies sport c
15			4 (188.RMI) samples spent calling getinder
17	-	4 (0.02%)	if NMRdata virtual_path[pos-3] == NMRdata_virtual_path[pos-2]  D12 = 0.0
18			clse 30 (36.14) samples spent calling optiones 41 (49.40) samples spent calling optiones 41 (49.40) samples spent calling optiones 41 (49.40) samples spent calling optionery
19		83 (0.33%)	12 (14.48%) samples spont calling getindes D12 = NMMdata.info[NMMdata.virtual_path[pos-3],NMMdata.virtual_path[pos-3],dist ond
1 (0.004	140%)	4 (0.02%)	2 (50.0%) samples spent calling matindes 1 (25.0%) samples spent calling matindes 1 (Mindsat_virtual_pathpos-3) == virtuallastPos
i2 i3			013 - 8.0 clase
4		51 (0.20%)	28 (54.9%) samples sport calling optimizes [5 (31.7%) samples sport calling optimizes [7 (13.7%) samples sport calling optimizes [03 = NMEdata.info[NMEdata.virtual_path[pos-3],virtualLastPos].dist ond
66 (0.004	140%)	3 (0.01%)	2 (66.5%) angles spent calling <u>setriodes</u> if WM6data.virtual_path[pos-2] == virtuallastFos
57			223 = 8.0 clus
1 (0.004	400)	36 (0.14%)	4 (11.1%) samples spent calling <u>optimeter</u> 4 (11.1%) samples spent calling <u>optimeter</u> 4 (11.1%) samples spent calling <u>optimeter</u> (75.8%) samples spent calling <u>optimeter</u>
0	-+070)		023 = NMRdata.info[NMRdata.virtual_path[gos-2],virtualLastfvs].dist
51			13 (68.42%) samples spent calling phonomonia 1 (5.26%) samples spent calling phonomonia
2		19 (0.08%)	5 (26.5.32%) samples spent calling (Mondargila c8, 58 = obondargile(D23,D24,D34)
	01%)	29 (0.11%)	26 (99.68%) sweples spent calling <u>storistonangle</u> cu, sw = qtoristonangle(DIZ,DI3,DI4,DI3,DI4,DI3,DI4,DI3)
i4 i5		5 (0.02%)	3 × 58°Cu 5 (100.00%) samples spent calling 2 b × 58°cu b × 58°cu
6	-	2 (0.0079%)	b = 50 tu
57 58			d = eBrcu 1f lewitroslPos
69 70			Q = sprod(Q_before, a, b, c, d) lastpos = pos
12			brask else  628 (62 79) under genet cillion groot
3		897 (3.54%)	880 (97.20) samples spont calling agged (6.675) samples point calling agged 1 (0.11%) samples spont calling agged Q_virtual = qproid(Q_before,a,b,c,d) 2 (108.08) samples spont calling agged
4 5	- 1	2 (0.0079%)	quol = rotopt(Q_virtual,034) # TODO: (Emerson) não conseguimos fazer o calculo abaixo da mesma forma que o classicBF?
6		13 (0.05%)	10 (76.92%) samples spent calling getinodes 1 (7.00%) samples spent calling = (51.530%) samples spent calling = (51.530%) samples spent calling =
			vs = qml, vi = ml, tens(pirtualistFos), -mol.stom(pirtualFos) x 2 (98.8%) samples pent calling :
7	-	4 (0.02%)	2 (S0.NM%) samples spent calling :  y = qmol.v2 + mol.stons[virtuallastFos].y-mol.stons[virtualFos].y
9	1	3 (0.01%)	v: = qmlvi = nd.items[virtualLastFos].z-mol.atoms[virtualFos].z 2 (06.57%) samples spent calling : 1 (33.33%) samples spent calling : 1 (vxvx + yvvy + vz*z > virtual_z² 4 (108.00%) samples spent calling model
80	- 4	4 (0.02%)	Q before = qurod(Q before, a, -b, -c, d) else
32			Q_before = Q_virtual end
4 5			pos = pos+1 end
36 37		4 (0.02%)	ed (418.48%) samples sport calling Indext
8			
		6302 (24.90%)	<pre>geol = rotopt(Q,DM) 68 (1.00) tamples speet calling settiments 88 (0.00) tamples speet calling settiments 8 (0.00) tamples speet calling settiment 10 (2.00) tamples 10 (2.00) tample</pre>
+			68 (1.88%) samples speet calling sationestry  608 (98.1%) supples speet calling sationestry  8 (0.18%) samples speet calling sationestry  18 (0.29%) samples speet calling sationestry  mol.atost[].element = NBMStat.ain([].].nval[[].aton[
		6302 (24.90%) 50 (0.20%)	GB (1.8%) implies speet calling materials (2.8%) implies speet calling statements (2.8%) implies (2.8%) implies (2.8%) implies (2.8%) implies (2.8%) implies (2.8%) implies speet calling statements (2.8%) implies speet calling statement (2.8%) impli
19		50 (0.20%)	GR (1.8%) implies speet calling Sationers's GR (0.5%) implies speet calling Sationers's R (0.1%) implies speet calling Sationers's R (0.1%) implies speet calling Sationers's R (0.1%) implies speet calling Sationers's R (0.2%) implies speet
			68 (1.88%) samples spent calling sationersty  58 (9.58%) supples spent calling sationersty  8 (0.38%) camples spent calling sationersty  18 (2.78%) samples spent calling sationersty  18 (2.78%) samples spent calling sationersty  19 (2.88%) samples spen
91	3	50 (0.20%)	68 (1.8%) samples speet calling satisfacestry  88 (89.15%) samples speet calling satisfacestry  18 (2.15%) samples speet calling satisfacestry  18 (2.15%) samples speet calling satisfacestry  18 (2.10%) samples speet calling satisfacestry  18 (2.0%) samples speet calling satisfacestry  28 (2.0%) samples speet calling satisfacestry  29 (2.0%) samples speet calling satisfacestry  29 (2.0%) samples speet calling satisfacestry  20 (3.0%) samples speet calling satisfacestry
1 2 3 97 (0.3	3	3 (0.01%)	68 (1.8%) samples spent calling gatamatrix  8 (0.1%) tamples spent calling gatamatrix  1 (0.1%) tamples spent calling gatamatrix  1 (0.1%) tamples spent calling gatamatrix  2 (1.0%) tamples spent calling structure  8 (0.0%) tamples spent calling structure  8 (0.0%) tamples spent calling structure  8 (0.0%) tamples spent calling structure  1 (0.0%) tamples spent ca
0 1 1 2 2 3 97 (0.3	1.38%)	3 (0.01%)	Get (1.873) samples speet calling statements (2.88 (0.5.173) samples speet calling statement (3.88 (0.5.173) samples speet calling statement (4.10.89) samples speet calling statement (5.10.89) sampl
0 1 1 2 2 3 97 (0.5 4 4 4 5 5 (1.05%)	: : : : : : : : : : : : : : : : : : : :	50 (0.20%) 3 (0.01%) 1140 (4.50%) 1786389 (7057.48%)	Get (1.8%) samples speet calling statements (1
2 2 3 97 (0.5 4 4 5 (1.05%) 6 6 7 1 (0.004)	:::::::::::::::::::::::::::::::::::::::	50 (0.20%) 3 (0.01%) 1140 (4.50%) 1786:389 (7057.48%)	### Comparison of the Comparis
0 0 1 1 2 2 2 3 3 97 (0.5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	:::::::::::::::::::::::::::::::::::::::	3 (0.01%) 3 (0.01%) 1140 (4.50%) 1786389 17(057,48%)	### Company of the Co
2 2 2 4 4 4 5 5 266 (1.05%) 5 5 7 1 (0.004 3 3 9 13 (0.0	:::::::::::::::::::::::::::::::::::::::	50 (0.20%) 3 (0.01%) 1140 (4.50%) 1786:389 (7057.48%)	### (# 1.11)   Ample   Sport
2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	: : : : : : : : : : : : : : : : : : :	3 (0.01%) 3 (0.01%) 1140 (4.50%) 1786389 17(057,48%)	### (# 1.00 cm) cm   cm   cm   cm   cm   cm   cm
0 0 1 1 2 2 2 3 3 3 97 (0.3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	: : : : : : : : : : : : : : : : : : :	3 (0.01%) 3 (0.01%) 1140 (4.50%) 1786:389 (7057:48%) 1 (0.0040%) 19 (0.08%)	## (1.50) supples speet calling gattered ## (1.50) supples speet calling gatte
0 0 1 1 1 2 2 2 3 3 3 97 (0.3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4		3 (0.01%) 3 (0.01%) 1140 (4.50%) 1786:389 (7057:48%) 1 (0.0040%) 19 (0.08%)	
90 91 92 93 97 (0.5 94 95 (1.05%) 96 97 (0.004 98	: : : : : : : : : : : : : : : : : : :	3 (0.01%) 3 (0.01%) 1140 (4.50%) 11786389 (7057.48%) 1 (0.0040%) 19 (0.08%) 13 (0.05%)	## 1
00 0 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	: : : : : : : : : : : : : : : : : : :	3 (0.01%) 3 (0.01%) 1140 (4.50%) 11786389 (7057.48%) 1 (0.0040%) 13 (0.05%) 250 (0.99%) 41 (0.16%)	## (19 Man Seales were calling statements ## (19 Man Seales were calling
00 0 10 11 12 12 12 13 3 3 97 (0.3 13 13 13 14 14 15 15 15 16 16 17 17 17 (0.004 18 18 18 18 18 18 18 18 18 18 18 18 18	: : : : : : : : : : : : : : : : : : :	1786.389 (7057.48%) 1 (0.040%) 1 (0.05%) 1 (0.05%) 1 (0.05%) 1 (0.05%)	(1 (10) supple spect calling gazzanzia   (10 (10) supple spect calling
0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	: : : : : : : : : : : : : : : : : : :	3 (0.01%) 3 (0.01%) 1140 (4.50%) 11786389 (7057.48%) 1 (0.0040%) 13 (0.05%) 250 (0.99%) 41 (0.16%)	### Command of the process of the pr
00 10 10 10 10 10 10 10 10 10 10 10 10 1	( 1.3.3.38%) ( 1.3.3.38%) ( 1.3.3.38%) ( 1.3.3.38%) ( 1.3.3.3.38%) ( 1.3.3.3.38%) ( 1.3.3.3.3.3.3.3.3.3.3.3.3.3.3.3.3.3.3.3	3 (0.01%) 3 (0.01%) 1140 (4.50%) 11786389 (7057.48%) 1 (0.0040%) 13 (0.05%) 250 (0.99%) 41 (0.16%)	C. (1.00) taughts quest calling statistics   C. (1.00) taughts q
10   11   12   12   13   14   14   15   15   16   16   16   17   17   17   17   17	: : : : : : : : : : : : : : : : : : :	3 (0.01%) 3 (0.01%) 1140 (4.50%) 1140 (4.50%) 11786389 (7057.48%) 1 (0.0040%) 19 (0.08%) 13 (0.05%) 41 (0.16%) 9 (0.04%) 30 (0.12%)	C. 1.00. 1

file: ///home/gphilippi/working/artigo COAP/Molecular Con...

81		4853566 (19174.96%)	4 samples spent calling manimentar 10206 (0.21% pages spent calling gasterminos): closure 3 samples spent calling gatidate 3 samples spent calling gatidate 1 samples spent calling gatidate 1 samples spent calling to the spent spent spent spent spent spent calling to the spent
-			
81	4		else
81	5		nsol = nsol+1
81	6 9 (0.04%)	9 (0.04%)	storage_mol[msol] = copy(mol)
81	7		return
81	8		end
81	9		end
82	0		end #closure
81 81 82 82 82	1		
82	2		_l, _pos, _mol, _q = initialization()
82		10206 (40.32%)	13206 (108.00) sumples spent calling <u>materials</u> (clause quaternios);mol,o)
82	4		
82 82 82 82	5		return nsol, storage_nol
82	6		
82	7		end #solver quaternion8P
0.2	0		