Steve Aigbe and Christoph F. Eick

COSC 4368: Fundamentals of Artificial Intelligence Spring 2022

Problem Set1 (Individual Tasks[[1]](#footnote-1) Centering on Search)

Second Draft



Fig. 1: Finding a Needle in a Large Haystack with Intelligent Search

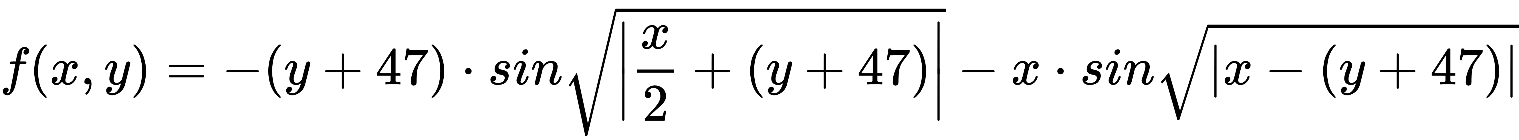
Submission Deadlines: Task1: Th., February 17, 11:59p; Task2: Mo., February 21, 11:59p

Last Updated: February 4, 8a

Weight: 30-34% of the points available for the 3 problem sets! Task weight: T1=31 points T2=35 points.

1) On Probabilistic Search Algorithms: Implementing and Experimenting with Randomized Hill Climbing *Steve*

Implement randomized hill climbing, called RHC in the following, and conduct a set of experiments, minimizing the following function f:



With -512£ x,y £512

Your procedure should be called RHC and have the following input parameters:

* sp: is the starting point[[2]](#footnote-2) of the Randomized Hill Climbing run
* p the number of neighbors of the current solution that will be generated
* z neighborhood size; for example if z is set to z=0.5 p neighbors for the current solution s are generated by adding vectors v=(z1,z2) with z1 and z2 being random numbers in [-0.5,+0.5] uniformly distributed
* seed which is an integer that will be used as the seed[[3]](#footnote-3) for the random generator you employ in your implementation.

RHC returns a vector (x,y) the value of f(x,y) and the number solutions that were generated during the run of RHC.

Run your randomized hill climbing procedure RHC twice[[4]](#footnote-4) for the following parameters:

sp = (404,504), (0,0.23), (-200,+300), (+412, -99.9)

p = 30 and 250

z = 3 and 0.5

For each of the 32 runs report:

a. the best solution (x,y) found and its value for f

b. number of solutions generated during the run[[5]](#footnote-5).

Summarize your results in 4 tables; one for each p and z combination[[6]](#footnote-6). Finally, run RHC one more time with “your preferred choice” of values for sp, p, z, and report the result; students, who find better solutions in this 33rd run will get more points for the 33rd run subtask. Interpret[[7]](#footnote-7) the obtained results evaluating solution quality, algorithm speed, impact of sp, p, and z on solution quality and algorithm speed. Do you believe with other values for p and r better results could be accomplished? At last, assess if RHC did a good, medium or bad job in computing a (local) minimum for f.

**Submission Guidelines:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| p=30 & z=0.5 | Run1 | | | Run2 | | |
| #sol se#sol searched  S s | sol | f(sol) | #sol  searched | Bes sol | F f(sol) |
| (2.9, (404,504) |  |  |  |  |  |  |
| (-2.5( (0,0.23) |  |  |  |  |  |  |
| ((4.2 (-200,+300) |  |  |  |  |  |  |
| (+412, -99.9) |  |  |  |  |  |  |

You should summarize your results in 4 tables formatted as the above, for each of the 4 combination of p & z. Don’t forget to summarize the results of your 33th run[[8]](#footnote-8) and to provide the other information asked for in the project specification!

**Failure to follow above instruction will lead to point deductions!**

2) Solving Discrete Constraint Satisfaction Problems TBDL

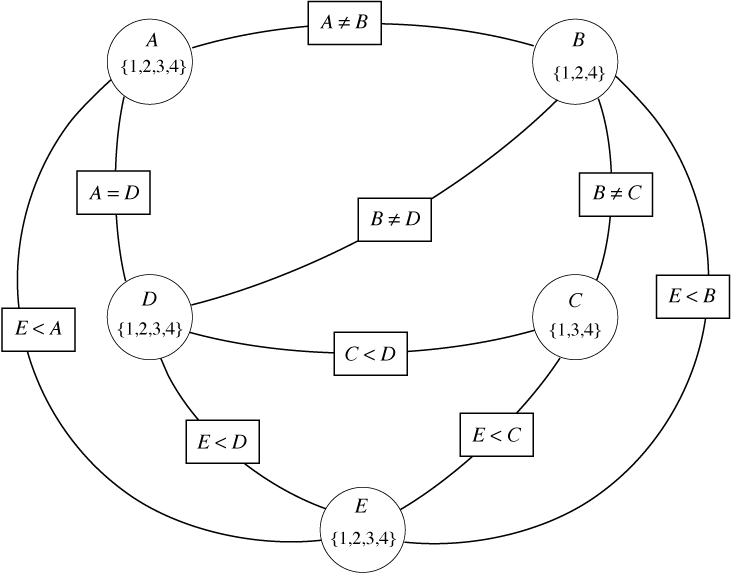


Fig. 2: Example of Constraint Graph

Write a program which finds solution to the following 3 hierarchically organized[[9]](#footnote-9) constraint satisfaction problems, involving 15 variables {A,B,C,…,N,O} which can take integer values in {1,…,100}.

1. Problem A: Find a solution to the constraint satisfaction problem involving the six variables A, B, C, D, E and F and constraints C1,…,C4:
   * (C1) A=B+C+E+F
   * (C2) D=E+F+21
   * (C3) D\*\*2=E\*E\*A + 694
   * (C4) E+F<A
2. Problem B: Find a solution to the constraint satisfaction problem involving ten variables A,…,J which satisfy constrains C1,…,C7:
   * (C5) H\*J+E\*16=(G+**I**)\*\*2 -48
   * (C6) A-C=(H-F)\*\*2+4
   * (C7) 4\*J=G\*\*2+7
3. Problem C: Find a solution to the constraint satisfaction problem involving 15 variables A,…,O which satisfy constrains C1,..,C13:
   * (C8) 2\*M=K\*\*2 25
   * (C0) (N-**O**)\*\*2 = (J-F)\***O**\*2
   * (C10) N\*\*2=M\*J+100
   * (C11) (L+N)\*\*2+1875=G\*(B+F)\*(K+M+N+30)
   * (C12) L\***O**=(A\*\*2)\*(K-G)
   * (C13) L\*\*3 = M\*\*2- (**O**\*F\*A)

Remark: In the above equations the letters ‘I’ and ‘O’ were put into bold face to avoid to be mistaken as numbers 0 or 1. Moreover, the letter ‘J’ looks somewhat similar than the letter ‘I’ but to better distinguish the two letters ‘J’ is never in bold face.

Your program should contain a counter **nva** (“number of variable assignments) that counts the number of times an initial integer value is assigned to a variable or the the assigned integer to the particular variable is changed; in addition to outputting the solution to the CSV also report the value of this variable at the end of the run, and an interface to call your program for CSP Problems A, B, or C. Your program should return the solution or “no solution exists” and the value of nva after the program terminates. Moreover, terminate the search as soon as you found a solution—do not search for additional solutions.

Submit a report which

* Gives a brief description of the strategy you used to solve the CSP
* Provides Pseudo Code of your CSP solver
* Explains the Pseudo Code in a paragraph or two
* Describes strategies (if you employed any) you employed to reduce the runtime of your program, measured by the final value of the variable nva.
* If you conducted a mathematical pre-analysis to eliminate variables, to obtain additional ‘<’ constraints to reduce search complexity or came up with other problem complexity reduction strategies based on such a pre-analysis, describe the results of the pre-analysis you conducted, and how the results of this pre-analysis were used for reducing the search complexity.
* If your program takes advantage of the hierarchical structure of the three CSP problems also explain how this was done.
* If the program you developed is generic in the sense that its code could be reused to solve constraint satisfactions which have a similar structure but different constraints, include a paragraph presenting evidence why your program has this property and what you did to make your program ‘generic’…

Moreover, submit the Source Code for the implementation in a separate file and instructions on how to run your code in a Readme File. Attach the Readme file as an appendix to your report.

Notes on grading:

* Sophisticated approaches that lead to lower complexities in solving the respective CSPs—measured by the final value of the variable nva—will get 10-15% higher scores compared to programs that use brute force approaches.
* Serious penalties will be assessed if the value of the variable nva is not properly computed.

1. Collaboration with other students is not allowed! [↑](#footnote-ref-1)
2. A vector (x,y) with x,y in [-512, +512] [↑](#footnote-ref-2)
3. If you run RHC with the same values for sp, p, z and seed it will always return the same solution; if you run is with the same values for sp, p, z and a different seed, it likely will return a different solution and the number of solutions searched is almost always different. [↑](#footnote-ref-3)
4. Make sure you use a different seed for your random generator to get a different sequence of random numbers for the 2 runs! [↑](#footnote-ref-4)
5. Count the number of times function f is called during the search! [↑](#footnote-ref-5)
6. (30, 0.5), (250, 0.5), (30, 3), (250, 3) [↑](#footnote-ref-6)
7. At least 25% of the available points will be allocated to interpreting the results. [↑](#footnote-ref-7)
8. Also briefly explain why you chose the particular input parameters for sp, p and z for your 33rd run! [↑](#footnote-ref-8)
9. A solution of the higher numbered problem also represents a solution of the lower numbered problem! [↑](#footnote-ref-9)