Mark Demore, 2d Lt

CSCE686 – Dr. Lamont

Project - Stochastic Design

**PROBLEM:**

Let *G* = (*V*, *A*) be a graph where *V* = {1, …, *n*} is a set of vertices representing air drop locations with the airfield located at vertex 1, and *A* is the set of arcs between them. Every arc (*i, j*) *i* =/=*j* is associated a non-negative cost matrix *C* = (*cij*). Let each vertex be assigned a set *S*,of *n* items, each with a weight *wi* and a value *vi*. Let P be a set of planes, each with a maximum cargo weight *Wi*. Constraints: (i) each air drop in *V*\{1} is visited exactly once by exactly one plane; (ii) all flight plans start and end at the airfield; (iii) . Objective: maximize total value and minimize total cost across all flight plans; *max* , *min*

1. ***Problem Domain Requirements Specification form:***

- domains, D

input Di - Graph G(X,Γ), X:locations. Γ: weighted vertex link set (cost); Set S(W,V), W: item weight, V: item value

output Do – Set of sets(R,L), R:route for each plane, L: load for each plane

- I(x); input conditions on input domain satisﬁed; x in X, link in Γ, set S

- O(x,z); output conditions on output/input domain satisﬁed; i.e.,

a feasible/optimal solution with respect to the input domain   
-- all x assigned  
-- max V (total value)

-- no *wp >* W (max weight)  
-- min C (total cost)

***Algorithm domain requirements specification form:***

* Name: stochastic-search genetic algorithm
* Domains: Ds is a set of satisfying solutions-a population; the population size n is the cardinality of Ds
* Operations:

I(x); x in Ds; x is a possible solution from population

O(x,z); x in Ds, z in Ds; z is a satisfying solution

***Algorithm domain design specification form:***

* Name: stochastic-search-ga(Ds)
* Domains: Di is set of algorithm-internal solutions, Ds is a set of satisfying solutions
* Imports: ADT set, list, real/integer/character
* Initialization of feasible solutions -> Ds; Di empty
* Operations I(x); x in Ds

O(x,z); x in Ds, z in Ds; condition on z being a satisfying solution

* Next-solution-generator -> x for x in Ds, Ds->Di
  + Recombination(crossover) x->y with crossover probability
  + Mutation x->y with mutation probability
  + Feasibility(y) -> Boolean [if true union(y,Di)] ‘genotype’
* Fitness/objective function mapping f(x) of each x in Di ‘phenotype’
* Selection Di -> Ds using f(x) as criteria, x in Di
* Axioms:

***Algorithm domain function specification form:***

* Function stochastic-search-ga(Ds)
* Initial condition: generate feasible Dinitial -> Ds, Di empty, pc, pm
* Body
  + - While not time/generation termination do ss-ga loop:
      * Next-state solution/population Ds, Ds->Di; do for each x in Ds, size n
        + Crossover(x) = y with pc
        + Mutation(x) = y with pm
        + If feasibility(y) then union (y, Di) -> Di
      * Fitness calculation f(x) for each x in Di
      * Selection(di) -> Ds based upon f(x), x in Di
    - End ss-ga while loop
    - Find optimal z in Ds
    - END function

***algorithm domain intermediate speciﬁcation form: (iterative)***

*• Heuristics: distance to next airdrop location, value of load item added*

*• Data structures: input – graph: set of nodes (locations), set of edge weight (cost between each location), set of items weight and value, set of planes with max weight; output – list of sets (route for each plane, and load for each plane)*

***algorithm domain function speciﬁcation form: (iterative)***

*•*  Function ss-ga(initial, Expand, Goal, Cost, Heuristic, crossover, mutation)

q <- New-Priority-Queue()

Insert (initial, q, Heuristic(initial))

**while** generation limit not reached

**do** current <- Extract-Min(q)

crossover(x) = y with pc

mutation(x) = y with pm

**if** Goal(current) then **return** solution

**for** each next in Expand(current)

**do** Insert (next, q, Cost(next) + Heuristic(next))

return failure

**References:**

[1] <https://www.geophysik.uni-muenchen.de/~igel/downloads/inviigenetic.pdf>

[2] <https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3>

[3] <https://en.wikipedia.org/wiki/Genetic_algorithm>

[4] <http://www.cs.ucc.ie/~dgb/courses/tai/notes/handout12.pdf>

[5] <https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html>

[6] [https://en.wikipedia.org/wiki/Knapsack\_problem](https://en.wikipedia.org/wiki/Knapsack_problem#:~:text=The%20knapsack%20problem%20is%20a,is%20as%20large%20as%20possible.)

[7] <https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/>

[8] <https://medium.com/@fabianterh/how-to-solve-the-knapsack-problem-with-dynamic-programming-eb88c706d3cf>

[9] <https://en.wikipedia.org/wiki/Vehicle_routing_problem>

[10] <https://developers.google.com/optimization/routing/vrp>

[11] <https://bib.irb.hr/datoteka/433524.Vehnicle_Routing_Problem.pdf>