Report LINGI2261: Assignment 4

Group N°11

Student1: Anh-Toan Le

Student2: Nils Boulanger

May 6, 2021

1 The bin packing problem (13 pts)

1. Formulate the bin packing problem as a Local Search problem (problem, cost function, feasible solutions, optimal solutions). **(1 pt)**

- **Problem**: find the best arrangement of items in order to use as few bins as possible and have the best possible arrangement in them.
- **Cost function**: the Fitness of the solution (to minimize).
- **Feasible solutions**: the bins are filled but their number is not minimized and they are not filled optimally with respect to the cost function.
- **Optimal Solutions**: solution that corresponds to a global minimum in the minimization of the cost function, i.e. solution for which the Fitness is minimum.

2. You are given a template on Moodle: *binpacking.py*. Implement your own extension of the *Problem* class from `aima-python3`. Implement the `maxvalue` and `randomized maxvalue` strategies. To do so, you can get inspiration from the *randomwalk* function in *search.py*. Your program will be evaluated on 15 instances (during 1 minute) of which 5 are hidden. We expect you to solve at least 12 out the 15.

   (a) `maxvalue` chooses the best node (i.e., the node with minimum value) in the neighborhood, even if it degrades the quality of the current solution. The `maxvalue` strategy should be defined in a function called *maxvalue* with the following signature: `maxvalue(problem,limit=100,callback=None)`. **(2.5 pts)**

(b) `randomized maxvalue` chooses the next node randomly among the 5 best neighbors (again, even if it degrades the quality of the current solution). The `randomized maxvalue` strategy should be defined in a function called *randomized_maxvalue* with the following signature: `randomized_maxvalue(problem,limit=100,callback=None)`. **(2.5 pts)**

3. Compare the 2 strategies implemented in the previous question and `randomwalk` defined in *search.py* on the given bin packing instances. Report, in a table, the computation time, the value of the best solution (in term of fitness) and the number of steps needed to reach the best result. For the randomized max value and the random walk, each instance should be tested 10 times to reduce the effects of the randomness on the result. When multiple runs of the same instance are executed, report the mean of the quantities. **(3 pts)**

	Max value			Random max value			Random walk		
Inst.	T(s)	Fit.	NS	T(s)	Fit.	NS	T(s)	Fit.	NS
i01	0.302	0.02565	8	0.571	0.02565	13.9	0.358	0.36845	59.7
i02	0.447	0.03163	9	0.503	0.03162	14.3	0.340	0.37349	71.5
i03	0.317	0.23114	4	0.554	0.23079	8.5	0.334	0.34290	72.1
i04	0.308	0.23370	5	0.578	0.23369	26.8	0.347	0.34753	52.5
i05	0.411	0.24730	1	0.886	0.24730	3.4	0.316	0.35613	73.1
i06	0.259	0.02784	5	0.546	0.02785	10.9	0.321	0.34866	57.7
i07	0.286	0.04172	3	0.564	0.04166	31.9	0.322	0.37280	61.3
i08	0.280	0.03700	3	0.560	0.03702	10.2	0.335	0.37115	87.6
i09	0.239	0.01431	4	0.420	0.01430	13.5	0.334	0.35423	73.3
i10	0.289	0.04324	4	0.669	0.04317	18.5	0.327	0.36765	76.5

NS: Number of steps — **T**: Time — **Fit.**: Fitness

4. **(4 pts)** Answer the following questions:

   (a) What is the best strategy? **(1 pt)**

Although the results in terms of Fitness are very close for the two strategies implemented here, the best one is `randomized max value`.

(b) Why do you think the best strategy beats the other ones? **(1 pt)**

Because `randomized max value` can avoid being in a local minimum. Moreover, it is a middle ground between intensification and diversification. When we run the algorithm several times, we can extract the best solution which is going to be better than the one found by `max value` if it was a local optimum.

(c) What are the limitations of each strategy in terms of diversification and intensification? **(1 pt)**

The strategy `max value` will maximize intensification while letting aside diversification. `random walk` will focus essentially on diversification, letting this time aside intensification. `Randomized max value` is a balance the two where we have both intensification and diversification.

(d) What is the behavior of the different techniques when they fall in a local optimum? **(1 pt)**

The strategy `max value` will continue to try to improve the solution but in vain. This explains the reduced number of steps before finding a solution. `randomized max value` can avoid falling in a local minima thanks to its random part. The ideal is to run the algorithm several times. The `random walk` will not care about that.

2 Propositional Logic (7 pts)

2.1 Models and Logical Connectives (1 pt)

Consider the vocabulary with four propositions A, B, C and D and the following sentences:

- $(\neg A \vee C) \wedge (\neg B \vee C)$
- $(C \Rightarrow \neg A) \wedge \neg(B \vee C)$
- $(\neg A \vee B) \wedge \neg(B \Rightarrow \neg C) \wedge \neg(\neg D \Rightarrow A)$

1. For each sentence, give its number of valid interpretations, i.e. the number of times the sentence is true (considering for each sentence **all the proposition variables** A, B, C and D). **(1 pt)**

- $(\neg A \vee C) \wedge (\neg B \vee C)$ has 5 valid interpretations
- $(C \Rightarrow \neg A) \wedge \neg(B \vee C)$ has 2 valid interpretations
- $(\neg A \vee B) \wedge \neg(B \Rightarrow \neg C) \wedge \neg(\neg D \Rightarrow A)$ has 1 valid interpretations

2.2 Color Grid Problem (6 pts)

1. Explain how you can express this problem with propositional logic. For each sentence, give its interpretation. **(2 pts)**

   - Each cell has to be colored : $\forall i, j \in E, \exists k \in E : C_{i,j,k}$ with $E = \{x \in \mathbb{N} | 0 \leq x < n = n\_rows\}$
   - Each row can only contain each color k once : $\forall i, j, j2, k \in E,$ with $j \neq j2 : C_{i,j,k} \Rightarrow \neg C_{i,j2,k}$
   - Each column can only contain each color k once : $\forall i, i2, j, k \in E,$ with $i \neq i2 : C_{i,j,k} \Rightarrow \neg C_{i2,j,k}$
   - Each diagonal can only contain each color k once :
     $\forall i, j, k \in E, \forall l \in \left[1,\ n\right[: C_{i,j,k} \Rightarrow \neg C_{(i+l) \bmod n,(j+l) \bmod n,k}$
   - Each anti-diagonal can only contain each color k once :
     $\forall i, j, k \in E, \forall l \in \left[1,\ n\right[: C_{i,j,k} \Rightarrow \neg C_{(i+l) \bmod n,(j-l) \bmod n,k}$
   - Each cell with an imposed color must be filled : $\forall$point $\in$ Points : C_{point}
     with Points $= \{(i, j, k) \in \mathbb{N} \mid (0, 0, 0) \leq (i, j, k) < (n, n, n)$ and given in input$\}$

2. Translate your model into Conjunctive Normal Form (CNF). **(2 pts)**

   - Each cell has to be colored : $\forall i, j \in E,\ \exists k \in E : C_{i,j,k}$ with $E = \{x \in \mathbb{N} \mid 0 \leq x < n\}$
   - Each row can only contain each color k once : $\forall i, j, j2, k \in E,$ with $j \neq j2 : \neg C_{i,j,k} \vee \neg C_{i,j2,k}$
   - Each column can only contain each color k once : $\forall i, i2, j, k \in E,$ with $i \neq i2 : \neg C_{i,j,k} \vee \neg C_{i2,j,k}$
   - Each diagonal can only contain each color k once :
     $\forall i, j, k \in E, \forall l \in \left[1,\ n\right[: \neg C_{i,j,k} \vee \neg C_{(i+l) \bmod n,(j+l) \bmod n,k}$
   - Each anti-diagonal can only contain each color k once :
     $\forall i, j, k \in E, \forall l \in \left[1,\ n\right[: \neg C_{i,j,k} \vee \neg C_{(i+l) \bmod n,(j-l) \bmod n,k}$
   - Each cell with an imposed color must be filled : $\forall$point $\in$ Points : C_{point}
     with Points $= \{(i, j, k) \in \mathbb{N} \mid (0, 0, 0) \leq (i, j, k) < (n, n, n)$ and given in input$\}$

3. Modify the function `get_expression(size)` in `cgp_solver.py` such that it outputs a list of clauses modeling the color grid problem for the given input. Submit your code on INGInious inside the *Assignment4: Color Grid Problem* task. The file `cgp_solver.py` is the *only* file that you need to modify to solve this problem. Your program will be evaluated on 10 instances of which 5 are hidden. We expect you to solve at least 8 out the 10. **(2 pts)**