Combinatorial Optimization, the Bin Packing Problem, and Constructive Heuristics

1 Before We Begin

Problem Instance Numbers: Later in this assignment, you will work through a few exercises, and each student's exercises will be different. Specifically, each student will use a different instance of the bin packing problem. When you get to the exercises in Sections 5, 6, 7, and 8, you will be asked to enter your problem instance number. This will be uniquely determined for each student by your student id number. It must be an integer, so start by dropping the Z. If there are any 0s at the start, then drop those as well. What remains is your problem instance number. For example, if your student id number is Z00123456789, then your problem instance number is 123456789. In the application that you are using the problem instance numbers are values of type long (in Java).

Session Logs: In the application that we will use, under the "Session" menu, you will find options to save a session log. This will generate a file with an extension of "*.ibp". You can otherwise name the file anything you'd like. If you work on the assignment over multiple sessions, just save each of the sessions. Make sure you don't select the same file as your previous session, or else it will be overwritten. Don't attempt to edit the session log files manually. Doing so will very likely generate alerts when your instructor views the session logs, possibly invalidating your work.

The session log files are what you will be submitting for this assignment, in addition to answers to any questions found later in this assignment. If you spread your work over multiple sessions, then just submit the log files for all of your sessions.

What Your Instructor Will be Looking for in the Session Logs: The instructor of your course, in addition to the answers to questions later in this document, will expect to find a record of the following in the session logs that you submit: (1) that you completed the "Default" problem instance in all 4 heuristic modes, when you worked through the complete tutorial in Section 4; and (2) that you completed your unique problem instance when you worked through the parts of the assignment from Sections 5, 6, 7, and 8.

2 Objectives, Prerequisites, and Time Requirements

This assignment will begin with a self-guided tutorial (Section 4) that will introduce you to combinatorial optimization. This includes learning about what combinatorial optimization problems are in general, as well as learning about a specific combinatorial optimization problem, the *Bin Packing Problem*, used by the tutorial to illustrate concepts such as lower bounds as well as how to use heuristics to quickly compute satisficing (though not necessarily optimal) solutions. The content of the tutorial is entirely within a Java application (instructions for downloading and running are found in Section 3).

After you finish working through the tutorial (Section 4), you will work through four exercises (Sections 5, 6, 7, and 8) to test your knowledge of the constructive heuristics that you learned about for the bin packing problem.

Objectives: The objectives of this assignment include:

- gaining a general understanding of combinatorial optimization problems;
- gaining an understanding of the bin packing problem, its applications, and how it is an example of a combinatorial optimization problem;
- learning about lower bounds;
- learning about constructive heuristics; and
- learning about the most common constructive heuristics for the bin packing problem, including first-fit, best-fit, first-fit decreasing, and best-fit decreasing.

Prerequisite Knowledge: The tutorial does not assume any prior knowledge of combinatorial optimization. However, it does assume that you have an understanding of introductory discrete mathematics, such as set theory and functions.

Technical Prerequisites: This assignment does not require programming. However, you will need to be able to run a Java application. The Java application in question requires Java 11 or later. You do not need a Java JDK as we will not be compiling the Java program. Only a Java Runtime Environment (JRE) is needed.

Time Requirements: The time required to complete this assignment including the time to work through the self-guided tutorial may vary by student depending upon prior background, course level, etc. We estimate that the time required is between 66 minutes and 135 minutes.

Note to Instructors: This assignment can potentially be used in courses on: (a) discrete mathematics toward the end to introduce combinatorial optimization, (b) algorithms to provide an example of an NP-Hard problem and how heuristics can sometimes be used to efficiently compute satisficing solutions despite the problem's complexity, or (c) artificial intelligence as an example of heuristic problem solving or just prior to coverage of local search algorithms.

3 Download Interactive Bin Packing (1–5 minutes)

In this assignment, we'll be using a Java application called *Interactive Bin Packing*. It is open source and can be found at this link: `https://github.com/cicirello/InteractiveBinPacking`. For this assignment, we won't need the source code of the application. Instead, we will use the jar file of the most recent release. Begin by downloading the *Interactive Bin Packing* application. Find instructions for downloading here: `https://github.com/cicirello/InteractiveBinPacking#installing-and-running-the-application`.

Note that in order to run the application you will need a Java Runtime Environment (JRE), version 11 or later. Since we will not be building from the source, you do not need a JDK—a JRE is sufficient. If you are not sure if you have Java installed, just try to run the application.

If you encounter any issues with the application, you are encouraged to report bugs and other issues via the issue tracker: `https://github.com/cicirello/InteractiveBinPacking/issues`.

4 Complete the Tutorial (45–90 minutes)

Before proceeding to specific bin packing problems, we'll start by working through a tutorial. At any point, you can stop and return later. Do the following:

1. Now that it has been downloaded, run the *Interactive Bin Packing* application. It is an executable jar file, so you can either just double click it, or you can run it from the command line. See this link if you need help running: `https://github.com/cicirello/InteractiveBinPacking#running`.
2. Click on *Info* menu and then the *Tutorial* option in that menu.
3. I recommend moving the Tutorial window that appears to the side of the application so that you can see both.
4. Read through the content of the Tutorial window in its entirety, while working along with it in the application. The tutorial will explain the Bin Packing problem, examples of a few applications of it, and then it will explain the most common constructive heuristics for the problem. I especially recommend using the application to follow along when you get to the heuristics, including switching the mode as indicated so that the application can help ensure that you are correctly understanding how the heuristics work.

Before moving on to the remainder of the assignment, this might be a good place to save your session log so that you have documentation that you completed the tutorial above. You can always save it again if you continue in this same session, or save an additional one if you take a break.

5 First-Fit Heuristic (5–10 minutes)

You will now compute the solution determined by the First-Fit heuristic for a specific bin packing instance. Do the following:

1. From the *Problem* menu, choose *Select Instance Number* to choose your unique problem instance number determined earlier in Section 1.
2. Compute the solution that would be found using the First-Fit heuristic. Use the application for guidance by switching the mode. It won't let you make mistakes.
3. Answer the following questions.

How many bins did this heuristic use?

What is the lower bound for this instance? Recall that the *Operations* menu has a command that will compute this for you.

Do we know if this solution is optimal? Do we have enough information to determine with certainty whether the solution found by this heuristic is the optimal solution (i.e., without doing any additional computation)? If yes, explain why. If no, explain why we don't know for sure.

6 First-Fit Decreasing Heuristic (5–10 minutes)

You will now compute the solution determined by the First-Fit Decreasing heuristic for a specific bin packing instance. Do the following:

1. From the *Problem* menu, choose *Select Instance Number* to choose your unique problem instance number determined earlier in Section 1.
2. Compute the solution that would be found using the First-Fit Decreasing heuristic. Use the application for guidance by switching the mode. It won't let you make mistakes.
3. Answer the following questions.

How many bins did this heuristic use?

What is the lower bound for this instance? Recall that the *Operations* menu has a command that will compute this for you.

Do we know if this solution is optimal? Do we have enough information to determine with certainty whether the solution found by this heuristic is the optimal solution (i.e., without doing any additional computation)? If yes, explain why. If no, explain why we don't know for sure.

7 Best-Fit Heuristic (5–10 minutes)

You will now compute the solution determined by the Best-Fit heuristic for a specific bin packing instance. Do the following:

1. From the *Problem* menu, choose *Select Instance Number* to choose your unique problem instance number determined earlier in Section 1.
2. Compute the solution that would be found using the Best-Fit heuristic. Use the application for guidance by switching the mode. It won't let you make mistakes.
3. Answer the following questions.

How many bins did this heuristic use?

What is the lower bound for this instance? Recall that the *Operations* menu has a command that will compute this for you.

Do we know if this solution is optimal? Do we have enough information to determine with certainty whether the solution found by this heuristic is the optimal solution (i.e., without doing any additional computation)? If yes, explain why. If no, explain why we don't know for sure.

8 Best-Fit Decreasing Heuristic (5–10 minutes)

You will now compute the solution determined by the Best-Fit Decreasing heuristic for a specific bin packing instance. Do the following:

1. From the *Problem* menu, choose *Select Instance Number* to choose your unique problem instance number determined earlier in Section 1.
2. Compute the solution that would be found using the Best-Fit Decreasing heuristic. Use the application for guidance by switching the mode. It won't let you make mistakes.
3. Answer the following questions.

How many bins did this heuristic use?

What is the lower bound for this instance? Recall that the *Operations* menu has a command that will compute this for you.

Do we know if this solution is optimal? Do we have enough information to determine with certainty whether the solution found by this heuristic is the optimal solution (i.e., without doing any additional computation)? If yes, explain why. If no, explain why we don't know for sure.

IMPORTANT: Make sure you save your session log to submit with your assignment.

9 Additional Questions

9.1 Iterative Sampling

In the course lectures, we saw that Iterative Sampling repeatedly generates N random solutions to a problem (without using a heuristic), and returns the best of those N random solutions. Although technically it doesn't use a heuristic, we can capture the essence of Iterative Sampling by using the First-Fit heuristic. How can we use the First-Fit heuristic to generate a series of N random solutions to a bin packing instance?

9.2 Stochastic Sampling with Heuristics

In the course lectures, we saw multiple stochastic sampling search algorithms that use a heuristic to bias the random decisions (e.g., VBSS and HBSS) while generating a sequence of N random solutions to the problem, returning the best of those N solutions. We could use either the First-Fit Decreasing or the Best-Fit Decreasing heuristic for this. Explain how either VBSS or HBSS would work if we were to use First-Fit Decreasing as the heuristic.

9.3 Hill-Climbing

In the course lectures, we saw different local search algorithms, including different hill climbers. In hill-climbing, you begin with a random solution to the problem. Then, a neighborhood operator is used to examine neighbors. Consider and answer the following questions.

9.3.1 Random Initial Solution

Describe how we can use the First-Fit heuristic to generate a random initial solution for our hill climber. If your answer to the earlier question 9.1 is correct, then you actually just need part of that answer here.

9.3.2 Neighborhood Operator

What can we use as our neighborhood operator? That is, given a current solution to the problem, how can we generate a random similar solution? Hint: Instead of directly rearranging items in the bins of the current solution, we can make a random change to the thing that led to the current solution, generating a neighbor of that.

10 What to Submit

For this assignment, you must submit the following:

- One or more Session Log files (with file extension "*.ibp") that proves that you completed the tutorial from part 4, specifically which shows that you worked through all four heuristics for the application's default bin packing instance.
- One or more Session Log files (with file extension "*.ibp") that proves that you completed the exercises from parts 5, 6, 7, and 8 for your unique problem instance based on your Stockton Z number. Note that this can be the same Session Log as the above item if you worked through everything during a single session.
- Your answers to the questions from parts 5, 6, 7, and 8 for your unique problem instance based on your Stockton Z number.
- Your answers to the questions from part 9.