

# Complex Task Allocation: A Comparison of Metaheuristics

Group 27

Lucas Wojciechowski, Ariel Weingarten, Alexander Maguire,  
Austin Dobrik, Dane Carr

July 21, 2014

## Summary

### 1 Introduction

### 2 Literature Review

### 3 Problem Formulation and Modeling

Suppose we have a number of mobile *robots*

$$robots = \{robot_i | i \in [1, n_{robots}]\}$$

Using these *robots*, we are to perform a number of tasks

$$tasks = \{task_i | i \in [1, n_{tasks}]\}$$

Each task has a *priority*

$$priority(task_i) \in [0, 1]$$

*robots* are not homogeneous; some are more “skilled” at certain tasks than others

$$skill(robot_i, task_j) \in [0, 1]$$

Tasks are located some distance each other

$$distance(task_i, task_j) > 0 \in \mathbb{R}$$

Each *robot* starts at some location, its *depot*.  $robot_i$  starts at  $depot_i$

Each *depot* is located a certain distance from each task.

$$distance(depot_i, task_j)$$

*tasks* are assigned to *robots*.

$$task_j \rightarrow robot_i$$

We seek, for each *robot*, a path through the *tasks* such that each *task* is performed exactly once and our overall cost is minimized.

A solution,  $S$ , consists of a mapping of *robots* to paths through *tasks*.

$$S(robot_i) = \{task_0, task_1, \dots, task_n\}$$

### 3.0.1 Cost Function

The cost of some path,  $S(i)$  is given by

$$cost(S(i)) = w(i, S(i, |S(i)|), S(i, 0)) + \sum_{t=0}^{|S(i)|-1} w(i, S(i, t), S(i, t+1))$$

Subject to the constraint

$$n \notin S(i) \quad \text{if} \quad M_{skill}(i, n) = 0$$

Therefore, the overall performance of a solution is:

$$\sum_{i \in robot} cost_i$$

Our weighted-distance function is defined as follows:

$$w(i, src, dst) = \begin{cases} (\alpha - priority(dst))^\gamma \times (\beta - skill(i, dst)) \times distance(src, dst) & dst \neq S(i, 0) \\ distance(src, dst) & \text{otherwise} \end{cases}$$

where  $\alpha, \beta \geq 1$ . These parameters are to ensure that the priority and skill factors are aligned with the minimization of the function.

$\gamma$  is used to tweak the relative importance of a task's priority, and has no constraints on it.

The  $distance(src, dst)$  function makes use of  $D_{task-task}$ ,  $D_{depot-task}$ , and  $D_{task-depot}$ .  $D_{task-task}$  is a square matrix that contains the distances between tasks i.e. the distance between task i and j is found at  $D_{task-task}(i, j)$ .  $D_{depot-task}$  contains the distances from depots to tasks i.e. the distance from depot i to task j is  $D_{task-task}(i, j)$ .  $D_{task-depot}$  contains the distances from the tasks to the depots i.e. the distance from task i to depot j is  $D_{task-depot}(i, j)$

The  $priority(dst)$  function makes use of  $P$ .  $P$  is a vector where the priority of  $task_i$  is contained in  $P(i)$ .

The  $skill(i, dst)$  function makes use of the matrix  $M_{skill}$ . The skill of  $robot_i$  at performing  $task_j$  is contained in  $M_{skill}(i, j)$

### 3.1 Modeling as Traveling Salesman Problem

Given a population of robots  $R$ , a set of tasks  $T$ , a priority for each task  $P$ , each robot's skill at performing each task  $M_{skill}(r, t)$  and an ordered assignment of tasks to robots  $S_r$ , for each  $r \in R$  we can create a directed, weighted graph as follows:

We will model complex task allocation loosely on the traveling salesman problem.

Given a random initial solution,  $S_0$ ,

$$\begin{aligned} G &= (V, E) \\ V &= \{start\} \cup T \\ E_{x,t} &= -P_a S_{r,t} \quad \forall t \in A_r \\ E_{x,start} &= 0 \end{aligned}$$

Which is to say, for each robot we create a graph with a start node and a node for each task assigned to said robot. The cost of traveling to any node is  $-P_a S_{r,t}$ . The cost of traveling to the start node is the distance from the current node to the start node.

From here, we can run TSP on the created graph, with the robot starting at the start node. It is trivial to show that regardless of the path taken, the total cost traveled is:

$$C_r = \sum_t^{A_r} -P_t S_{r,t}$$

In order to maximize efficiency of the system as a whole, we must then minimize the total cost for all robots:

$$C = \sum_r^R C_r = \sum_r^R \sum_t^{A_r} -P_t S_{r,t}$$

## 4 Potential other model

This is almost verbatim out of the literature.

$$x_{ij} = \begin{cases} 1 & \text{if arc}(i,j) \text{ is used in the tour} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$\text{minimize } \sum_{i=1}^n \sum_{j=1}^n c_{ij} \times x_{ij} \quad (2)$$

$$\sum_{i=1}^n \sum_{j=1}^n x_{ij} = n \quad (3)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |\text{SubTour}| - 1 \quad (4)$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A \quad (5)$$

$$\forall \text{SubTour} \subseteq V \setminus \phi \quad (6)$$

#### 4.1 Reduced Size Problem

In this section, we define a reduced size complex task allocation problem which will be used to perform hand-iterations of various optimization algorithms in the following sections.

Suppose we have 2 robots

$$\text{robots} = \{\text{robot}_0, \text{robot}_1\}$$

These robots must collectively perform 3 tasks

$$\text{tasks} = \{\text{task}_0, \text{task}_1, \text{task}_2\}$$

The priority of each task is

$$\begin{array}{ll} \text{task}_0 & 0.1 \\ \text{task}_1 & 0.2 \\ \text{task}_2 & 0.3 \end{array}$$

The skill of each robot at each task is

	$\text{task}_0$	$\text{task}_1$	$\text{task}_2$
$\text{robot}_0$	0.1	0.4	0.5
$\text{robot}_1$	0.2	0.3	0.6

The distances between tasks are

	$\text{task}_0$	$\text{task}_1$	$\text{task}_2$
$\text{task}_0$	0	0.1	0.2
$\text{task}_1$	0.1	0	0.3
$\text{task}_2$	0.2	0.3	0

Each robot,  $robot_i$ , starts at a depot,  $depot_i$ . The distance from each depot to each task is

	$task_0$	$task_1$	$task_2$
$depot_0$	0.6	0.3	0.1
$depot_1$	0.5	0.4	0.2

As an initial solution, we will use

$$S = [0, 1, sentinel, 2]$$

Which specifies the paths

$$\begin{aligned} robot_0 : & depot_0 \rightarrow task_0 \rightarrow task_1 \rightarrow depot_0 \\ robot_1 : & depot_1 \rightarrow task_2 \rightarrow depot_1 \end{aligned}$$

## 4.2 Real Problem

# 5 Proposed Solution

## 5.1 Taboo Search Algorithm

## 5.2 Simulated Annealing Algorithm

## 5.3 Genetic Algorithm

## 5.4 Particle Swarm Optimization Algorithm

## 5.5 Ant Colony Optimization Algorithm

# 6 Performance Evaluation

# 7 Conclusions & Recommendations

# References