Reaction Functions for Task Allocation to Cooperative Agents*

Xiaoming Zheng Sven Koenig
University of Southern California
Computer Science Department
Los Angeles, California, USC
{xiaominz,skoenig}@usc.edu

ABSTRACT

In this paper, we present ARF, our initial effort at solving task-allocation problems where cooperative agents need to perform tasks simultaneously. An example is multi-agent routing problems where several agents need to visit targets simultaneously, for example, to move obstacles out of the way cooperatively. First, we propose reaction functions as a novel way of characterizing the costs of agents in a distributed way. Second, we show how to approximate reaction functions so that their computation and communication times are polynomial. Third, we show how reaction functions can be used by a central planner to allocate tasks to agents. Finally, we show experimentally that the resulting task allocations are better than those of other greedy methods that do not use reaction functions.

Categories and Subject Descriptors

1.2 [**Artificial Intelligence**]: Problem Solving

General Terms

Algorithms

Keywords

Agent Coordination, Coalition Formation, Teamwork

1. INTRODUCTION

In recent years, several researchers have studied task-allocation problems in a competitive setting where agents are self-interested and try to maximize their own utilities [6, 9]. These task-allocation problems are then often solved with techniques from game theory [4, 1]. We are interested in general task-allocation problems in a cooperative setting where the agents collaborate to minimize the team cost (that is, maximize the team performance). In this paper, our motivating problem is multi-agent routing because it is very visual. The tasks are to visit targets in the plane. The terrain, the locations of all homogeneous agents and the locations of all targets are known.[1] We need to determine which targets each agent should visit and when it should visit them so that the team cost (for example, the amount of energy needed by the team or the task-completion time) is as small as possible. Multi-agent routing is a standard problem for robot teams, for example, as part of de-mining, search-and-rescue and taking rock probes on the moon. Multi-agent routing where each target needs to be visited by exactly one agent is currently a standard test domain for robot coordination with auctions [3]. In this paper, we extend multi-agent routing to the case where each target needs to be visited simultaneously by a given number of agents, which can be different for each target. Some targets need to be visited by only one agent. However, some targets now need to be visited by several agents simultaneously. For example, large fires can only be extinguished with several fire engines, and heavy objects can only be moved with several robots. Thus, the agents have to solve complex scheduling problems where the cost of visiting a target depends not only on the target and the agents that visit it but also on the visit time, which is different from the assumptions made by existing approaches for task allocation via coalition formation [7, 11]. We introduce our new approach, called ARF, as follows: After describing multi-agent routing problems formally, we first propose reaction functions as a novel way of characterizing the costs of agents in a distributed way. Second, we show how one can approximate reaction functions so that their computation and communication times are polynomial. Third, we show how a central planner can use reaction functions to allocate targets to agents and determine their visit times with a computation time that is exponential in the largest coalition size and polynomial in the number of agents and targets. Finally, we show experimentally that the target allocations of ARF are better than those of other greedy methods that do not use reaction functions.

2. MULTI-AGENT ROUTING

We now formalize multi-agent routing problems: The finite set of agents is A. The finite set of targets is X. The number of different agents that need to visit target x simultaneously, called its **coalition size**, is $d(x)$.[2] We call a target x **simple** if $d(x) = 1$

*This research was partly supported by NSF awards under contracts ITR/AP0113881, IIS-0098807, IIS-0350584 and IIS-0413196 as well as seed funding from NASA's Jet Propulsion Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, companies or the U.S. government.

Cite as: Reaction Functions for Task Allocation to Cooperative Agents, Xiaoming Zheng and Sven Koenig, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

Copyright © 2008, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

[1] One can solve multi-agent routing problems in unknown terrain by making assumptions about the unknown terrain, such as the assumption that it is traversable, making it in effect "known" and thus solvable with our task-allocation algorithms. One then runs the task-allocation algorithms again to re-allocate all unvisited targets to agents whenever this assumption turns out to be wrong and thus needs to get revised.

[2] For any complex target x with coalition size $d(x)$, the central planner needs to evaluate all possible coalitions of $d(x)$ different

and **complex** otherwise. The set of simple targets and the set of complex targets partition the set of all targets. We distinguish these two kinds of targets because an agent can freely determine when to visit simple targets but needs to agree with other agents on when to visit complex targets. The set of $d(x)$ different agents that need to visit complex target x at some visit time t is called the **coalition** for the complex target. Each agent in the coalition thus has a **commitment** to visit the complex target x at visit time t, written as $x \leftarrow t$. An **allocation** of agent a consists of a pair (X_a, C_a), where X_a is the set of simple and complex targets assigned to it and C_a is the set of its commitments for the complex targets. We say that the multi-agent routing problem is one with **disjoint coalitions** if every agent can visit at most one complex target because coalitions then cannot overlap. Otherwise, we say that the multi-agent routing problem is one with (potentially) **overlapping coalitions**. Disjoint coalitions are important in the presence of capacity constraints. For example, a fire engine that has only enough water to help extinguish one fire can visit only one complex target. However, it might be able to transport survivors to various hospitals and thus visit several simple targets. The **agent cost** $c_a^{agent}(X_a, C_a)$ is the smallest sum of travel and wait times needed for agent a to visit all targets X_a assigned to it, where it can freely determine when to visit each simple target subject to the restriction that it has to visit all complex targets (if any) at the visit times recorded in its commitment set C_a. (The agent cost is infinity in case the agent cannot satisfy this restriction.) Our objective is to find a solution with a small team cost, where a solution requires each target x to be assigned to exactly $d(x)$ different agents so that all agents in the coalition have the same commitment for a complex target. In addition, every agent can be assigned at most one complex target for multi-agent routing problems with disjoint coalitions. We consider two different ways of defining the team cost. The **team cost** is $\sum_{a \in A} c_a^{agent}(X_a, C_a)$ (roughly proportional to the energy needed by the agents for moving and waiting) for the **MiniSum team objective** and $\max_{a \in A} c_a^{agent}(X_a, C_a)$ (the task-completion time) for the **MiniMax team objective**. We use c^{team} as a special operator for either the sum or max operator, depending on the team objective, and write $c_{a \in A}^{team}\, c_a^{agent}(X_a, C_a)$ to make our notation independent of the team objective.

3. OUR APPROACH

Our approach to multi-agent routing consists of two stages. In Stage 1, all simple targets are assigned to agents. In Stage 2, all complex targets are assigned to agents. Stages 1 and 2 both consist of multiple rounds, where one additional target is myopically assigned to some agent (for simple targets) or a coalition of agents (for complex targets) during each round, namely under the assumption that it is the last target to be assigned. In particular, one additional target is assigned to some agent or coalition of agents so that the team cost after the assignment is smallest among all possible assignments, resulting in a form of hill-climbing. Thus, we expect the team cost of the resulting solution to be small. The simple targets are assigned before the complex targets so that the agents can manipulate the order in which they visit the simple targets assigned to them to accommodate the complex targets assigned to them. (The opposite would not work since complex targets are assigned to agents with a visit time that the agents need to obey while simple targets are assigned to them without a visit time.) For multi-agent routing problems with disjoint coalitions, an agent is no longer assigned any targets once it has been assigned one complex target.

agents and there are an exponential number of them. To make the central planner fast, we assume that the largest coalition size of all complex targets is small. Fortunately, this assumption often holds in practice.

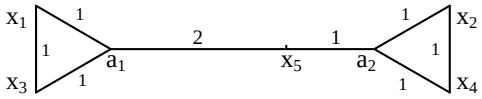

Figure 1: Multi-Agent Routing Problem on a Graph

3.1 Stage 1: Assigning Simple Targets

The central planner assigns one additional simple target to some agent during each round of Stage 1 until all simple targets have been assigned. There are several methods in the literature for allocating simple targets to agents. We assign them to agents with sequential single-item auctions [12] because we are able to reuse the principle underlying sequential single-item auctions when assigning complex targets to agents in Stage 2. In the beginning of Stage 1, $X_a = \emptyset$ and $C_a = \emptyset$ for all agents a. Consider any round of Stage 1. The central planner needs to assign one additional simple target to some agent so that the resulting team cost is smallest. Each agent a uses the Or-opt heuristic [8], as given in the appendix, to approximate for each unassigned simple target x (that is, simple target not assigned to any agent) its agent cost $\mathcal{F}_a^x := c_a^{agent}(X_a \cup \{x\}, \emptyset)$ for visiting target x and all (simple) targets already assigned to it at the optimal visit times. Each agent a then submits for each unassigned simple target x the value

$$\mathcal{V}_a^x := \begin{cases} \mathcal{F}_a^x - c_a^{agent}(X_a, \emptyset) & \text{for MiniSum} \\ \mathcal{F}_a^x & \text{for MiniMax} \end{cases}$$

to the central planner. Let X_s be the set of unassigned simple targets. The central planner determines $(a, x) := \arg\min_{a \in A, x \in X_s} \mathcal{V}_a^x$ and then assigns simple target x to agent a (which executes $X_a := X_a \cup \{x\}$), terminating the round. It has been shown that this assignment results in the smallest team cost among all possible assignments of any unassigned simple target to any agent [12], resulting in a form of hill-climbing. The procedure then repeats until all simple targets have been assigned to agents.

Example: Consider the multi-agent routing problem shown in Figure 1 for the MiniMax team objective, where the agents and targets are located on a graph and the agents can only move along the edges of the graph. x_1, x_2, x_3 and x_4 are simple targets. x_5 is the only complex target with $d(x_5) = 2$. It takes four rounds to allocate the simple targets in Stage 1. In the first round, $\mathcal{F}_{a_1}^{x_1} = \mathcal{F}_{a_1}^{x_3} = \mathcal{F}_{a_2}^{x_2} = \mathcal{F}_{a_2}^{x_4} = 1$ and $\mathcal{F}_{a_1}^{x_2} = \mathcal{F}_{a_1}^{x_4} = \mathcal{F}_{a_2}^{x_1} = \mathcal{F}_{a_2}^{x_3} = 4$. The central planner can make one of the following four allocations since $\mathcal{V}_a^x = \mathcal{F}_a^x$ for the MiniMax team objective: x_1 to a_1, x_3 to a_1, x_2 to a_2 or x_4 to a_2. Assume that the central planner always breaks ties in favor of the target with the smallest index value and thus allocates x_1 to a_1. Then, in the second round $\mathcal{F}_{a_1}^{x_2} = \mathcal{F}_{a_1}^{x_4} = 6$, $\mathcal{F}_{a_2}^{x_3} = 4$, $\mathcal{F}_{a_1}^{x_3} = 2$ and $\mathcal{F}_{a_2}^{x_2} = \mathcal{F}_{a_2}^{x_4} = 1$. The central planner thus allocates x_2 to a_2. In a similar way, the central planner allocates x_3 to a_1 in the third round and x_4 to a_2 in the fourth round, which terminates Stage 1. ■

3.2 Stage 2: Assigning Complex Targets

The central planner assigns one additional complex target to some coalition of agents during each round of Stage 2 until all complex targets have been assigned. The central planner proceeds for Stage 2 in a way similar to Stage 1. In the beginning of Stage 2, X_a and C_a are as they were at the end of Stage 1. Consider any round of Stage 2. The central planner needs to assign one additional complex target to some coalition of eligible agents and determine its visit time so that the resulting team cost is smallest. For multi-agent routing problems with disjoint coalitions, only

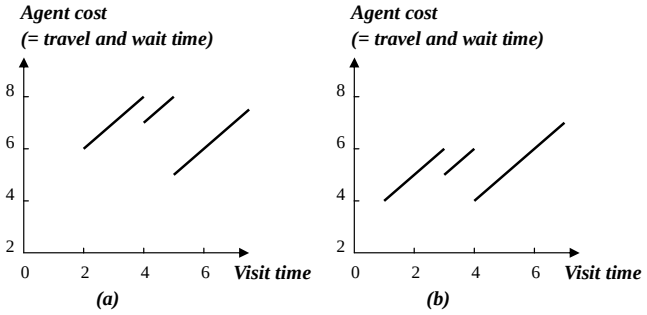

Figure 2: Reaction Functions of a_1 (left) and a_2 (right)

those agents are **eligible** to get targets assigned that have not yet been assigned a complex target. For multi-agent routing problems with overlapping coalitions, all agents are eligible to get targets assigned. Each eligible agent a determines for each unassigned complex target x (that is, complex target not assigned to any coalition of agents) and each visit time t of target x its agent cost $\mathcal{F}_a^x(t) := c_a^{agent}(X_a \cup \{x\}, C_a \cup \{x \leftarrow t\})$ for visiting target x at visit time t, all simple targets already assigned to it at the optimal visit times and all complex targets already assigned to it at the visit times recorded in its commitment set. We view the $\mathcal{F}_a^x$ as **reaction functions** that map t to $\mathcal{F}_a^x(t)$. Each eligible agent a then submits for each unassigned complex target x the function

$$\mathcal{V}_a^x(t) := \begin{cases} \mathcal{F}_a^x(t) - c_a^{agent}(X_a, C_a) & \text{for MiniSum} \\ \mathcal{F}_a^x(t) & \text{for MiniMax} \end{cases}$$

to the central planner. Let $P(n)$ be the set that contains all sets of n different eligible agents and X_c be the set of unassigned complex targets. The central planner determines $(P, x, t) := \arg\min_{P \in P(d(x)), x \in X_c, 0 \le t < \infty} c_{a \in P}^{team} \mathcal{V}_a^x(t)$ and then assigns complex target x to all agents $a \in P$ with visit time t (which execute $X_a := X_a \cup \{x\}$ and $C_a := C_a \cup \{x \leftarrow t\}$), terminating the round. The procedure then repeats until all complex targets have been assigned to coalitions of agents.

Theorem 1 *Assigning complex target x to all $d(x)$ agents $a \in P$ with visit time t (as determined by the central planner) results in the smallest team cost among all possible assignments of any unassigned complex target x' to any $d(x')$ different eligible agents with any visit time.*

Proof: For the MiniSum team objective, $c_{a \in P}^{team} \mathcal{V}_a^x(t) = \sum_{a \in P} (\mathcal{F}_a^x(t) - c_a^{agent}(X_a, C_a))$ is the increase in the sum of the agent costs of the agents in the coalition. Thus, it is also the increase in the sum of the agent costs of all agents since no other agent is assigned a target. Let c be the sum of the agent costs of all agents before the assignment. Then, $c_{a \in P}^{team} \mathcal{V}_a^x(t) + c$ is the sum of the agent costs of all agents after the assignment. Minimizing $c_{a \in P}^{team} \mathcal{V}_a^x(t)$ thus also minimizes the sum of the agent costs of all agents (= the team cost for the MiniSum team objective) after the assignment. For the MiniMax team objective, $c_{a \in P}^{team} \mathcal{V}_a^x(t) = \max_{a \in P} \mathcal{F}_a^x(t)$ is the largest agent cost after the assignment of the agents in the coalition. Let c be the largest agent cost of any agent before the assignment. Since the assignment of a target to an agent cannot decrease its agent cost, $\max(c_{a \in P}^{team} \mathcal{V}_a^x(t), c)$ is the largest agent cost of any agent after the assignment. Minimizing $c_{a \in P}^{team} \mathcal{V}_a^x(t)$ thus also minimizes the largest agent cost of any agent (= the team cost for the MiniMax team objective) after the assignment. ■

Example: Consider again the multi-agent routing problem shown in Figure 1 for the MiniMax team objective. It takes one round to allocate complex target x_5 in Stage 2. Figure 2 shows the reaction functions of both agents for target x_5 in this round. We briefly discuss the reaction function of agent a_1 for target x_5: Agent a_1 can visit target x_5 (and wait there for the other agent to visit the target simultaneously) and then both of its simple targets x_1 and x_3 (Order 1). Agent a_1 can also visit one of its simple targets, then target x_5 (and wait there for the other agent to visit the target simultaneously) and finally its other simple target (Order 2). Agent a_1 can also visit both of its simple targets and then x_5 (and wait there for the other agent to visit the target simultaneously). Let t be the visit time of x_5. First, if $t < 2$, then agent a_1 cannot visit target x_5. Thus the first segment of the reaction function is a constant segment of infinite value starting at $(0, \infty)$. More and more orders become available as the visit time t increases since each available order remains available. Second, if $2 \le t < 4$, then the agent can visit target x_5 only with Order 1. Its reaction function is the sum of its travel time for Order 1 (= 6) plus its wait time at target x_5 (= $t - 2$) in [2,4). Thus, the second segment of the reaction function is a linear segment with slope one starting at $(2, 6)$. Third, if $4 \le t < 5$, then agent a_1 can visit target x_5 with both Orders 1 and 2. Order 2 results in the smallest sum of travel and wait times. Its reaction function is the sum of its travel time for Order 2 (= 7) plus its wait time at target x_5 (= $t - 4$) in $[4, 5)$. Thus, the third segment of the reaction function is a linear segment with slope one starting at $(4, 7)$. Finally, if $5 \le t$, then agent a_1 can visit target x_5 with Orders 1, 2 and 3. Order 3 results in the smallest sum of travel and wait times. Its reaction function is the sum of its travel time for Order 3 (= 5) plus its wait time at target x_5 (= $t - 5$) in $[5, \infty)$. Thus, the last segment of the reaction function is linear segment with slope one starting at $(5, 5)$. The central planner allocates target x_5 to both agents with visit time 5 since $\mathcal{V}_a^x(t) = \mathcal{F}_a^x(t)$ for the MiniMax team objective, which terminates Stage 2. ■

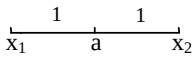

Figure 3: Multi-Agent Routing Problem on a Graph

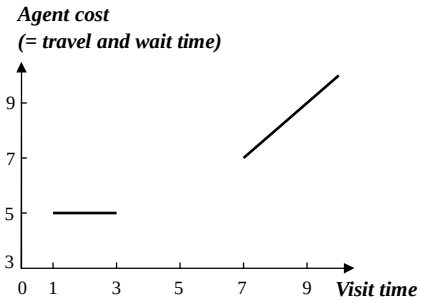

Figure 4: Reaction Function of a

4. APPROXIMATIONS

In general, the reaction function of agent a for complex target x is a piecewise linear function with segments of three different kinds:

- Type 1: constant segments of infinite value, modeling the case where agent a cannot visit all complex targets already assigned to it at the visit times recorded in its commitment set if it visits target x at the given visit times (since there does not exist a solution);
- Type 2: constant segments of finite values, modeling the case where the optimal solution is for agent a to visit at least one complex target already assigned to it after target x if it visits target x at the given visit times (since waiting at target x then does not increase its agent cost); and

- Type 3: linear segments with slope one, modeling the case where the optimal solution is for agent a to visit target x after all complex targets already assigned to it (if any) if it visits target x at the given visit times (since waiting at target x then increases the agent cost by the wait time).

Example: Figure 3 presents part of a different multi-agent routing problem to illustrate the three kinds of segments, where the agents and targets are located on a graph and the agents can only move along the edges of the graph. x_1 and x_2 are complex targets. Assume that complex target x_1 has been assigned to agent a with visit time 5 in the first round of Stage 2. Figure 4 then shows the reaction function of agent a for target x_2 in the second round of Stage 2. Let t be the visit time of target x_2. First, if $t < 1$ then agent a cannot visit target x_2 since it needs a travel time of one from its current location to target x_2. Thus, the first segment of the reaction function is a constant segment of infinite value starting at $(0, \infty)$ (Type 1). Second, if $1 \leq t \leq 3$ then the optimal solution is for agent a to visit target x_2 before target x_1. A wait time at target x_2 does not change the agent cost of 5 since the agent visits target x_1 at visit time 5 and then stops. Thus, the second segment of the reaction function is a constant segment of finite value starting at $(1, 5)$ (Type 2). Third, if $3 < t < 7$ then agent a cannot visit target x_2 since it visits target x_1 at visit time 5 and needs a travel time of two from target x_1 to target x_2. Thus, the third segment of the reaction function is a constant segment of infinite value starting at $(3, \infty)$. (Type 1). Finally, if $7 \leq t$ then the optimal solution is for agent a to visit target x_1 before target x_2. A wait time at target x_2 increases the agent cost 7 by the wait time since the agent reaches target x_2 at time 7, waits for other agents to visit the target simultaneously and then stops. Thus, the last segment of the reaction function is a linear segment with slope one starting at $(7, 7)$ (Type 3). ■

Each eligible agent a has to solve an optimization problem for each visit time t of each unassigned complex target x to determine its reaction function for the target because it needs to determine the optimal way of visiting the simple targets so that it visits complex target x at visit time t and all complex targets already assigned to it at the visit times recorded in its commitment set. The resulting reaction functions are complex, which makes their computation and communication time-intensive. Therefore, we now present a method that calculates approximate reaction functions, basically by only considering a constant number of orders of the simple targets instead of all of them.

4.1 Disjoint Coalitions

We discuss multi-agent routing problems with disjoint coalitions first. In this case, the reaction function of agent a for complex target x is a piecewise linear function with segments of Types 1 and 3 only. Segments of Type 2 are ruled out (since they require the agent to be assigned at least one complex target already), which simplifies our method for the approximation of the reaction function. Our method approximates the reaction functions in two different ways: First, it discretizes the possible visit times of complex target x into time intervals. Second, it uses the Or-opt heuristic [8] to approximate the optimal solution for each time interval, that is, the solution with the smallest agent cost where agent a visits complex target x in the time interval without waiting and all simple targets already assigned to it at the optimal visit times. Let the visit time of target x be t and the resulting agent cost be c. Our method then fixes the resulting order of visiting all targets and increases the visit time of target x by making the agent wait at it for other agents to visit the target simultaneously, resulting in an **interval function** with a constant segment of infinite value starting at $(0, \infty)$ and a linear segment with slope one starting at (t, c). Our method then calculates the approximate reaction function as the minimum of all interval functions. These approximate reaction functions have a constant number of segments because our method considers only a constant number of orders of visiting all targets, which is a simplification. However, our method would calculate the true reaction functions if it used an infinite number of time intervals and determined the truly optimal solutions for all time intervals. We now give a step by step explanation of our method:

- Step 1: Our method finds the smallest visit time s of complex target x if agent a visits the complex target before all of its simple targets. This visit time is equal to the travel time from the current location of the agent to the complex target. Then, our method uses the Or-opt heuristic [8] to approximate the smallest visit time e of complex target x if the agent visits the complex target after all of its simple targets.

- Step 2: Our method divides the time interval $[s, e]$ evenly into k time intervals $(s_i, e_i]$ for a given discretization granularity k and considers these time intervals together with the time interval $[0, s]$. (Our method actually uses the time interval $(s_0 = -\epsilon, e_0 = s]$ for a small positive constant ϵ instead so that all time intervals are open on the left and closed on the right.)

- Step 3: For each $0 \leq i \leq k$, our method finds the agent cost of agent a for visiting complex target x in time interval $(s_i, e_i]$ without waiting and all simple targets already assigned to it at the optimal visit times. This optimization problem is a special case of the NP-hard traveling salesperson problem with time windows [2]. Our method uses a version of the Or-opt heuristic [8], as given in the appendix, to solve it approximately. Let the visit time of target x be $t_i \in (s_i, e_i]$ and the resulting agent cost be c_i. (It holds that $t_0 = e_0$.) Our method then defines the following interval function that calculates the agent cost if all targets are visited in the given order and the agent waits $t - t_i \geq 0$ time units at target x for other agents to visit the target simultaneously:

$$\mathcal{F}^x_{a,i}(t) := \begin{cases} \infty & \text{if } 0 \leq t < t_i \\ c_i + t - t_i & \text{if } t_i \leq t. \end{cases}$$

- Step 4: Our method determines the approximate reaction function of agent a for complex target x as the minimum of the interval functions $\mathcal{F}^x_{a,i}$ for all $0 \leq i \leq k$ since each interval function expresses the agent cost if agent a visits its simple targets and the complex target in a particular order:

$$\mathcal{F}^x_a(t) := \min_{0 \leq i \leq k} \mathcal{F}^x_{a,i}(t).$$

This minimization can be performed in polynomial time as follows since $t_0 < \ldots < t_k$: Start a new constant segment of the approximate reaction function with infinite value at $(0, \infty)$ unless $t_0 = 0$. Set $t = 0$ and $c = \infty$. For each $i = 0 \ldots k$ do: If $c + t_i - t \leq c_i$ then do nothing (since the current segment goes through $(t_i, c + t_i - t)$ and thus is no larger than the linear segment with slope one starting at (t_i, c_i)) and go on to the next i. Otherwise, start a new linear segment of the approximate reaction function with slope one starting at (t_i, c_i). Set $t = t_i$ and $c = c_i$ and go on to the next i. The resulting approximate reaction function thus has at most $k + 2$ segments (including the initial constant segment of infinite value, if any) and can be represented with a constant amount of memory and be communicated in constant time.

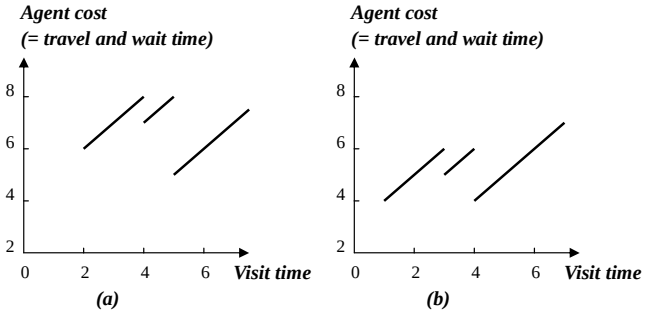

Figure 5: Approximate Reaction Functions of a_1 (left) and a_2 (right) for Discretization Granularity $k = 3$

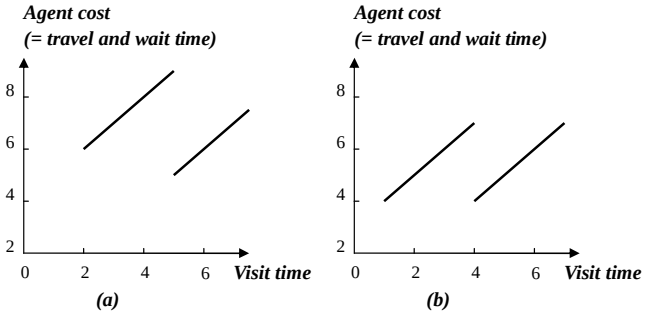

Figure 6: Approximate Reaction Functions of a_1 (left) and a_2 (right) for Discretization Granularity $k = 2$

Example: Consider again the multi-agent routing problem shown in Figure 1. The approximate reaction functions of both agents in the only round of Stage 2 are shown in Figure 5 for discretization granularity $k = 3$. They consist of four segments each (including the initial constant segment of infinite value) and are identical to the true reaction functions. The approximate reaction functions of both agents in the only round of Stage 2 are shown in Figure 6 for for discretization granularity $k = 2$. They consist of three segments each and are no longer identical to the true reaction functions. ■

4.2 Overlapping Coalitions

We now discuss multi-agent routing problems with overlapping coalitions. In this case, the reaction function of agent a for complex target x is a piecewise linear function with segments of Types 1, 2 and 3. We thus need to generalize our method for the approximation of the reaction function from the previous section:

- Step 1: This step is identical to Step 1 for multi-agent routing problems with disjoint coalitions, except that our method approximates the smallest visit time e of complex target x if the agent first visits all complex targets already assigned to it at the visit times recorded in its commitment set, then all simple targets, and finally target x.

- Step 2: This step is identical to Step 2 for multi-agent routing problems with disjoint coalitions.

- Step 3: For each $0 \leq i \leq k$, our method uses the Or-opt heuristic [8] to approximate the agent cost of agent a for visiting complex target x in time interval $(s_i, e_i]$ without waiting and all simple targets already assigned to it at the optimal visit times. We call this order the calculated order. Our method then defines the interval function $\mathcal{F}^x_{a,i}(t)$ as follows:

  1. Start a constant segment of infinite value at $(0, \infty)$. There are no constraints.
  2. Consider the solution where the agent visits each simple target as early as possible so that it visits target x and the simple targets in the calculated order, it visits the complex targets already assigned to it at the visit times recorded in its commitment set, and it obeys all constraints. (This can simply be done by maintaining a Queue 1 that contains target x and the simple targets in the calculated order and a Queue 2 that contains the complex targets in the calculated order. If the agent can visit the first target in Queue 1 and then the first target in Queue 2 at the visit time recorded in its commitment set and there is no constraint that the agent needs to visit the first target in Queue 1 after the first target in Queue 2, then let the agent visit the first target in Queue 1 next, delete the target from Queue 1, and repeat. Otherwise, let the agent visit the first target in Queue 2 next, delete the target from Queue 2, and repeat.) Let the visit time of target x in this solution be t and the resulting agent cost be c.
  3. If the agent does not visit at least one complex target already assigned to it at some point in time after target x in the above solution, then start a linear segment with slope one at (t, c) and stop.
  4. Let the first complex target that the agent visits at some point in time after target x in the above solution be x' and its visit time be t'. Let the travel time from target x (possibly via simple targets) to target x' in the above solution be Δt. Start a constant segment of finite value at (t, c). Set t to $t' - \Delta t$.
  5. If the agent visits target x directly before target x', then start a constant segment of infinite value at (t, ∞), add the constraint that the agent needs to visit target x at some point in time after target x' and goto 2.
  6. Otherwise, the agent visits a simple target directly before target x'. Add the constraint that the agent needs to visit this simple target at some point in time after target x' and goto 2.

  The number of segments of the interval function is linear in the number of iterations of the above algorithm since each iteration adds only one or two segments to the interval function. The number of iterations is linear in the number of constraints added since each iteration adds at most one constraint. Finally, the number of constraints added is at most quadratic in the number of targets since the constraints are of the form "visit a given simple target after a given complex target." Thus, the number of segments of each interval function is at most quadratic in the number of targets. Only the last segment of each interval function is a linear segment with slope one.

- Step 4: Our method determines the approximate reaction function of agent a for complex target x as the minimum of the interval functions $\mathcal{F}^x_{a,i}$ for all $0 \leq i \leq k$:

$$\mathcal{F}^x_a(t) := \min_{0 \leq i \leq k} \mathcal{F}^x_{a,i}(t).$$

  This minimization can be performed in polynomial time as follows: Set t to zero. Determine the segment of all interval functions whose function value is smallest at time point

t, breaking ties in favor of constant segments with finite values. Start a new segment of the approximate reaction function of the same kind as this segment at the point given by t and the function value of this segment. If this segment is a constant segment of finite or infinite value, then advance t to the smallest time point where a new segment starts for any interval function, and repeat the procedure until no new segment starts for any interval function. If this segment is a linear segment with slope one, then determine the segment of all interval functions whose function value is smallest at time point t among all constant segments of finite value. Advance t to the minimum of the intersection point and the smallest time point where a new segment starts for any interval function, and repeat the procedure until no new segment starts for any interval function.

The number of segments of each interval function is at most quadratic in the number of targets. Thus, the number of segments of all interval functions is also quadratic in the number of targets since there are only a constant number of interval functions. The above algorithm can add additional segments, possibly one for each intersection that a linear segment with slope one has with some other segment. The number of linear segments with slope one is one for each interval function. Thus, their total number is constant and the number of intersections that they have with other segments is at most linear in the number of segments. The resulting approximate reaction function thus has a number of segments that is at most quadratic in the number of targets and can thus be represented with a polynomial amount of memory and be communicated in polynomial time.

5. WINNER DETERMINATION

In Stage 1, each agent submits the value $\mathcal{V}_a^x$ for each unassigned simple target x to the central planner. The central planner determines $(a, x) := \arg\min_{a \in A, x \in X_s} \mathcal{V}_a^x$ with a computation time that is linear in the product of the number of agents and the number of targets and then assigns simple target x to agent a. In Stage 2, each eligible agent calculates its approximate reaction function $\mathcal{F}_a^x(t)$ for each unassigned complex target x and then submits the function

$$\mathcal{V}_a^x(t) := \begin{cases} \mathcal{F}_a^x(t) - c_a^{agent}(X_a, C_a) & \text{for MiniSum} \\ \mathcal{F}_a^x(t) & \text{for MiniMax} \end{cases}$$

to the central planner. The central planner determines $(P, x, t) := \arg\min_{P \in P(d(x)), x \in X_c, 0 \leq t < \infty} c_{a \in P}^{team} \mathcal{V}_a^x(t)$ and then assigns complex target x to all agents $a \in P$ with visit time t. The minimization is over $0 \leq t < \infty$. The following theorem, however, shows that this is unnecessary.

Theorem 2 *Let $T(P, x)$ be the set of time points that correspond to the starting points of all constant segments with finite values and all linear segments with slope one of the functions $\mathcal{V}_a^x(t)$ for all eligible agents $a \in P$ and unassigned complex targets x. Then,*

$$\min_{P \in P(d(x)), x \in X_c, 0 \leq t < \infty} c_{a \in P}^{team} \mathcal{V}_a^x(t) = \min_{P \in P(d(x)), x \in X_c, t \in T(P,x)} c_{a \in P}^{team} \mathcal{V}_a^x(t).$$

Proof by contradiction: Assume that the equality does not hold and consider any coalition $P \in P(d(x))$ and target $x \in X_c$. Let $t \notin T(P, x)$ be the time point that minimizes the left-hand side, which is then strictly smaller than the right hand side since it minimizes over a larger set of time points. $c_{a \in P}^{team} \mathcal{V}_a^x(t)$ cannot increase when decreasing t by an infinitesimal amount since none of the functions $\mathcal{V}_a^x$ can increase unless $t \in T(P, x)$. Thus, one can decrease t until $t \in T(P, x) \cup \{0\}$ without increasing $c_{a \in P}^{team} \mathcal{V}_a^x(t)$, which is a contradiction. Thus, one needs to minimize only over $t \in T(P, x) \cup \{0\}$ rather than $0 \leq t < \infty$. The team cost at $t = 0$ is infinity unless $0 \in T(P, x)$. Thus, one needs to minimize only over $t \in T(P, x)$ rather than $t \in T(P, x) \cup \{0\}$. (This is also the reason why the set $T(P, x)$ of time points does not need to include the starting points of all constant segments with infinite value.) ■

Example: Consider again the multi-agent routing problem shown in Figure 1. The central planner needs to minimize only over the time points 1, 2, 3, 4 and 5 in the only round of Stage 2 for discretization granularity $k = 3$. ■

The computation time of

$$\min_{P \in P(d(x)), x \in X_c, t \in T(P,x)} c_{a \in P}^{team} \mathcal{V}_a^x(t)$$

and thus the computation time of the central planner in each round of Stage 2 is linear in the product of the number of coalitions of $d(x)$ different eligible agents (with an upper bound that is linear in the number of agents to the power of the largest coalition size), the number of unassigned complex targets (with an upper bound that is linear in the number of targets) and the number of segments of the functions $\mathcal{V}_a^x(t)$ (with an upper bound that is constant for multi-agent routing problems with disjoint coalitions and quadratic in the number of targets for multi-agent routing problems with overlapping coalitions). Overall, the computation time of the central planner is linear in the product of the computation time in each round and the number of rounds (with an upper bound that is linear in the number of targets) and thus exponential in the largest coalition size and polynomial in the number of agents and targets.

6. EXPERIMENTAL RESULTS

We now evaluate our method, that we call ARF (= Approach with Reaction Functions) in the following. There are no other task-allocation methods for multi-agent routing with complex targets. In order to demonstrate the benefits of reaction functions in our approach, we compare ARF against two other greedy task-allocation methods that do not use reaction functions.

1. **Greedy 1:** Greedy 1 is similar to ARF except that it mixes the allocation of simple and complex targets in one stage and restricts agents to visit their targets in the order in which they were assigned to them, similar to [10]. Greedy 1 consists of one stage with multiple rounds, where one additional target (either a simple target or a complex target) is assigned per round with a given visit time to some agent until all targets have been assigned. Each additional target is assigned to some agent with a given visit time so that the team cost after the assignment is smallest among all possible assignments under the restriction that agents visit their targets at the agreed-on visit times. Consider any round. Let X_a be the set of targets already assigned to agent a in previous rounds. Let c_a be its agent cost for visiting all targets already assigned to it at the agreed-on visit times. Each eligible agent a submits for each unassigned target x the value $\mathcal{V}_a^x$ to the central planner. For multi-agent routing problems with disjoint coalitions, all agents are eligible to get target x assigned if x is a simple target but only those agents are eligible to get target x assigned that have not yet been assigned a complex target if x is a complex target. For multi-agent routing problems with overlapping coalitions, all agents are eligible to get target x assigned. $\mathcal{V}_a^x$ is the

Agents	Number of Simple Targets	Number of Complex Targets	MiniSum Team Objective						MiniMax Team Objective					
			Greedy 1		Greedy 2		ARF		Greedy 1		Greedy 2		ARF	
			Team Cost	Computation Time	Team Cost	Computation Time	Team Cost	Computation Time	Team Cost	Computation Time	Team Cost	Computation Time	Team Cost	Computation Time
4	8	2	417.6	0.60	417.2	0.10	359.8	1.33	130.9	0.90	139.5	0.40	129.5	0.67
4	18	2	558.3	3.80	555.2	7.70	457.1	13.40	161.3	3.20	170.9	2.50	156.4	3.80
4	28	2	656.2	7.60	668.2	6.59	519.0	103.20	185.3	6.50	190.3	19.00	174.6	24.00
6	7	3	434.8	1.10	436.1	0.60	401.9	1.20	109.2	0.50	113.4	0.10	108.5	0.20
6	17	3	575.4	3.00	572.9	3.40	495.7	5.40	134.2	2.60	134.8	1.40	126.8	3.60
6	27	3	658.1	6.90	654.4	25.20	565.8	43.30	151.4	7.60	150.4	8.90	142.9	12.90
8	6	4	455.9	0.60	457.9	0.20	438.1	0.90	102.0	0.80	103.1	0.30	102.0	0.90
8	16	4	582.9	4.40	589.7	1.80	537.3	6.20	123.8	4.10	120.1	1.00	118.4	5.67
8	26	4	680.6	9.20	674.8	15.60	603.9	30.10	133.8	8.50	136.8	6.00	129.7	8.10
10	5	5	471.0	0.60	473.6	0.60	459.6	1.00	95.3	1.20	94.8	0.40	94.4	1.33
10	15	5	596.1	3.60	601.8	2.60	544.6	5.90	112.5	3.80	112.8	1.50	110.6	5.20
10	25	5	696.1	9.50	694.1	11.00	621.1	19.10	123.5	10.70	124.5	6.10	118.1	16.67

Table 1: Experimental Results for Multi-Agent Routing Problems with Disjoint Coalitions

Agents	Number of Simple Targets	Number of Complex Targets	MiniSum Team Objective						MiniMax Team Objective					
			Greedy 1		Greedy 2		ARF		Greedy 1		Greedy 2		ARF	
			Team Cost	Computation Time	Team Cost	Computation Time	Team Cost	Computation Time	Team Cost	Computation Time	Team Cost	Computation Time	Team Cost	Computation Time
4	35	5	757.1	23.40	744.0	28.70	688.8	249.10	228.0	14.30	222.6	44.20	202.8	144.30
4	30	10	909.1	45.10	895.6	32.10	869.4	279.00	269.8	16.30	272.4	25.80	254.5	215.70
4	20	20	1053.3	15.70	1042.9	17.30	1021.7	155.40	306.9	17.20	319.6	12.80	298.7	167.60
4	0	40	1020.9	16.40	1020.9	11.10	1020.9	22.00	295.3	18.70	295.0	14.70	295.3	15.80
6	35	5	672.8	22.50	661.8	24.60	617.5	215.10	153.5	14.60	156.5	22.20	141.6	45.20
6	30	10	818.3	24.50	808.9	26.40	779.6	172.90	183.1	16.30	181.8	16.20	167.4	54.10
6	20	20	960.9	15.50	962.6	13.30	950.2	100.90	210.3	17.40	210.9	12.30	200.9	41.00
6	0	40	969.8	17.00	969.8	12.40	969.8	18.90	216.3	19.10	216.4	17.70	216.4	15.10
8	35	5	605.7	15.50	608.9	46.40	548.4	172.80	115.5	13.30	110.0	19.10	100.3	25.10
8	30	10	733.5	15.30	729.8	29.50	692.5	128.50	137.2	15.70	130.6	14.60	122.4	26.70
8	20	20	898.6	17.60	899.5	11.90	882.9	96.30	162.8	17.80	160.9	12.20	151.4	23.40
8	0	40	930.6	18.00	930.6	13.30	930.6	19.00	172.9	19.90	172.9	16.00	172.9	16.40
10	35	5	555.6	14.30	548.8	33.10	496.0	100.10	91.3	15.20	86.8	12.00	81.9	19.80
10	30	10	681.5	15.80	675.7	19.20	635.4	90.30	110.4	14.30	106.3	19.20	102.8	20.40
10	20	20	838.5	18.40	844.9	10.30	820.9	61.40	129.8	16.60	126.8	12.40	123.8	18.20
10	0	40	904.8	19.90	904.8	14.50	904.8	21.00	143.7	19.90	143.3	16.70	143.7	17.70

Table 2: Experimental Results for Multi-Agent Routing Problems with Overlapping Coalitions

agent cost of agent a for visiting all targets already assigned to it at the agreed-on visit times and target x last and without waiting. Thus, $\mathcal{V}_a^x = c_a + t(a, x)$, where $t(x, a)$ is the travel time from the location of agent a after it visited all targets in X_a at the agreed-on visit times to target x. Let X' be the set of unassigned targets. The central planner determines $(P, x) := \arg\min_{P \in P(d(x)), x \in X'} \max_{a \in P} \mathcal{V}_a^x$ for the MiniMax team objective and $(P, x) := \arg\min_{P \in P(d(x)), x \in X'} \sum_{a \in P} (\max_{a' \in P} \mathcal{V}_{a'}^x - c_a)$ for the MiniSum team objective, and then assigns target x to agent(s) $a \in P$ with visit time $\max_{a \in P} \mathcal{V}_a^x$. The procedure then repeats until all targets have been assigned to agents.

2. **Greedy 2:** Greedy 2 is a cross between Greedy 1 and ARF. It consists of two stages with multiple rounds, where one additional target is assigned per round to some agent. In Stage 1, all simple targets are assigned to agents in the same way as done by ARF. In Stage 2, all complex targets are assigned to agents in the same way as done by Greedy 1. That is, all agents visits their complex targets (if any) last.

We evaluate ARF, Greedy 1 and Greedy 2 for multi-agent routing problems on known four-neighbor planar grids of size 51×51 with square cells that are either blocked or unblocked. The grids resemble office environments with randomly closed doors from [5]. All complex targets x have coalition size $d(x) = 2$. ARF uses discretization granularity $k := min(20, 2|X_a|)$, where X_a is the set of simple targets assigned to agent a in Stage 1. We vary the number of agents from 4, 6, 8 to 10. For multi-agent routing problems with disjoint coalitions, we vary the number of targets from 10, 20 to 30 and set the number of complex targets to half the number of agents, so that every agent visits exactly one complex target. For multi-agent routing problems with overlapping coalitions, we fix the number of targets at 40 and vary the number of complex targets from 5, 10, 20 to 40. For each of these scenarios, we average over 100 runs with randomly generated cells for the agents and targets. Table 1 presents the average computation times of the three task-allocation methods (in milliseconds) and the resulting average team costs for multi-agent routing problems with disjoint coalitions. Table 2 presents the same results for multi-agent routing problems with overlapping coalitions.

We make the following observations: While the computation times of ARF are generally larger than those of Greedy 1 and Greedy 2 in both tables, their order of magnitude is about the same (except for a small number of cases). In particular, ARF is able to solve large multi-agent routing problems in real time. For multi-agent routing problems with disjoint coalitions, the team costs of Greedy 1 and Greedy 2 are about the same, while ARF reduces the team cost significantly, namely by about 11-12 percent for the MiniSum team objective and 3-5 percent for the MiniMax team objective. For multi-agent routing problems with overlapping coalitions, ARF also reduces the team cost significantly when the number of complex targets is small. However, the advantage of ARF diminishes as the number of complex targets increases. In particular, when all 40 targets are complex ones, the team costs of ARF, Greedy 1 and Greedy 2 are about the same because ARF allows the agents to manipulate the order in which they visit the simple targets assigned to them to accommodate the complex targets assigned to them. If there are no simple targets, then ARF, Greedy 1 and Greedy 2 degenerate to the same method. We attempted to find the smallest team costs for the multi-agent routing problems by

formulating them as mixed integer programs, seeding the solutions with the allocations obtained by ARF, and then solving the mixed integer programs with CPLEX. However, CPLEX was unable to improve on the team costs found by ARF within our time limit of three hours.

7. CONCLUSIONS

In this paper, we extended multi-agent routing to the case where targets need to get visited simultaneously by several agents. We proposed ARF, that uses reaction functions as a novel way of characterizing the costs of agents in a distributed way. We showed how to approximate reaction functions so that their computation and communication times are polynomial. We showed how reaction functions can be used by a central planner to allocate targets to agents and determine their visit times with a computation time that is exponential in the largest coalition size and polynomial in the number of agents and targets. It is future work to experiment with different approximation and hill-climbing schemes. For example, our hill-climbing method assigns the complex targets from easy to difficult, and we intend to experiment with the reverse order. It is also future work to extend our approach to generalizations of multi-agent routing problems. For example, reaction functions can also be used if the agents are heterogeneous and the complex targets need to be visited by different kinds of agents. In this case, each agent calculates only the reaction functions for the complex targets that it can visit, and the central planner evaluates only coalitions of suitable agents for visiting complex targets. Finally, it is future work to extend our approach to task-allocation problems other than multi-agent routing.

8. REFERENCES

[1] V. Dang and N. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 564–571, 2004.

[2] J. Desrosiers, Y. Dumas, M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter 2, pages 35–139. North-Holland, 1995.

[3] B. Dias and A. Stentz. Opportunistic optimization for market-based multirobot control. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 2714–2720, 2002.

[4] J. Kahan and A. Rapoport. *Theories of coalition formation*. Lawrence Erlbaum Associates, 1984.

[5] S. Koenig, C. Tovey, X. Zheng, and I. Sungur. Sequential bundle-bid single-sale auction algorithms for decentralized control. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1359–1365, 2007.

[6] S. Kraus, O. Shehory, and G. Taase. Coalition formation with uncertain heterogeneous information. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1–8, 2003.

[7] H. Lau and L. Zhang. Task allocation via multi-agent coalition formation: Taxonomy, algorithms and complexity. In *Proceedings of the International Conference on Tools with Artificial Intelligence*, pages 346–350, 2003.

[8] I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Northwestern University, Evanston, Illinois, 1976.

[9] T. Sandholm and V. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1-2):99–137, 1997.

[10] S. Sariel and T. Balch. Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In *Proceedings of the AAAI-05 Workshop on Integrating Planning into Scheduling*, pages 27–33, 2005.

[11] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998.

[12] C. Tovey, M. Lagoudakis, S. Jain, and S. Koenig. The generation of bidding rules for auction-based robot coordination. In L. Parker, F. Schneider, and A. Schultz, editors, *Multi-Robot Systems: From Swarms to Intelligent Automata*, pages 3–14. Springer, 2005.

9. APPENDIX

We stated that our method uses a version of the Or-opt heuristic [8] to approximately determine the agent cost c of agent a and the visit time t for visiting complex target x in time interval $(s_i, e_i]$ without waiting and all simple targets already assigned to it at the optimal visit times. This version of the Or-opt heuristic is given in the following. $Evaluate(S)$ simply inserts complex target x at all positions into S and calculates the travel time needed for agent a to visit all targets in the given order, including complex target x (without waiting). It then returns the pair of the travel time and visit time of complex target x, minimized over all cases (if any) where the visit time of complex target x falls into time interval $(s_i, e_i]$. If there are no such cases, it returns infinity for the travel time. Let S be a random permutation of the simple targets $x_1 \dots x_p$ already assigned to agent a in Stage 1. The Or-opt heuristic then takes all subsequences of lengths one to $p/2 + 1$ of S and inserts them in both their original and reserved order into all positions in S until the travel time returned by $Evaluate(S'')$ for the resulting permutation S'' is smaller than the travel time returned by $Evaluate(S)$. It then sets S to the resulting permutation and repeats the process.

```
1  S := (x_{n(1)} ... x_{n(p)}) (Randomly);
2  (c, t) := Evaluate(S);
3  for q := p/2 ... 0 do
4      for r := 1 ... p − q do
5          S' := S;
6          remove (x_{n(r)} ... x_{n(r+q)}) from S';
7          for each position s in S' do
8              S'' := insert (x_{n(r)} ... x_{n(r+q)}) at position s into S';
9              (c', t') := Evaluate(S'');
10             if c' < c then
11                 S := S''; c := c';
12                 break loops and go to line 4;
13             end
14             S'' := insert (x_{n(r+q)} ... x_{n(r)}) at position s into S';
15             (c', t') := Evaluate(S'');
16             if c' < c then
17                 S := S''; c := c';
18                 break loops and go to line 4;
19             end
20         end
21     end
22 end
23 (c, t) := Evaluate(S);
24 return (c, t);
```