Task allocation for multi-robot teams in dynamic environments

Maciej Hojda

Wroclaw University of Science and Technology,
Faculty of Computer Science and Management,
27 Wyb. Wyspianskiego St., 50-370 Wroclaw, Poland
maciej.hojda@pwr.edu.pl

Abstract. Multi-robot task allocation is a well known and widely researched decision-making problem that is difficult to solve in reasonable time even for small instances. Additional complexity is added by the fact that the parameters of the system may change over time, which happens either by external stimuli or by the task execution itself. One of the common causes behind these changes is the movement of executors or tasks. This paper tackles a problem of multi-task, multi-robot allocation in a such an environment. Formulated and solved is a specific decision-making problem. Performed is an experimental comparison of a dedicated solution algorithm with known methods for the more general Multidimensional Knapsack and Covering problem. Empirical evaluation illustrates that a dedicated approach is competitive and often necessary, as the general approach proves to be too slow.

Keywords: multi-robot task allocation, mobile routing, multidimensional knapsack and covering problem

1 Introduction

Multi-robot systems are expected to quickly and efficiently solve a wide variety of problems. Robots (or executors) are assigned to tasks for execution, and while complex tasks require cooperation of many robots, those executors which are more capable can perform multiple tasks on their own. Autonomous cooperating robots cover a wide spectrum of applications, including: data collection, inspection, monitoring, production and military use [CV,CM,Ho1,TI]. Particularly, of growing interest are problems when tasks are spatially distributed and movement is required in order to perform them [Ho2,LLFNS,MPA,RSSH,TLLB]. Furthermore, the executors are often equipped with multiple tools, such as multiple means of communication [JS,YMHC] or can be selected to perform tasks with varying levels of intensity [MPA,SBCFTW]. Varying modes of execution allow to fine-tune the execution process through more efficient use of available resources. As the tasks and executors grow more diverse, so increases the complexity of single-robot control and management of robotic teams. This paper focuses on the latter aspect by providing a method of joint task allocation and routing in a spatially distributed multi-robot environment.

W. Mitkowski et al. (Eds), *Trends in Advanced Intelligent Control, Optimization and Automation*,
Advances in Intelligent Systems and Computing 577, DOI 10.1007/978-3-319-60699-6_47

Formulated and solved is a problem of multi-robot task allocation with mobile executors. The goal is to assign robots to tasks and determine the order of task execution for each executor. In [Ho1] this specific problem was proven to be NP-hard even when formulated in its feasibility version. The approach was to solve a substitutive problem instead. This paper improves upon that result by adapting the solution algorithm to solve the stated problem directly. A comparison with known solution algorithms for the more general Multidimensional Knapsack and Covering Problem [AHL,HALG] emphasises the benefits of a dedicated solution algorithm, what is visible in both the execution time and the quality of the obtained solution.

This paper is divided into five sections: the introduction, the problem formulation, the solution algorithm presentation, the empirical evaluation and the conclusions.

2 Problem formulation

There are I executors, J tasks and L modes, indexes of which are given as follows: executors $\mathbf{I} \triangleq \{1, 2, \ldots, I\}$, tasks $\mathbf{J} \triangleq \{1, 2, \ldots, J\}$, modes $\mathbf{K} \triangleq \{1, 2, \ldots L\}$. Decision variable defines the assignment of tasks to executor and the order of execution and is denoted by $x \triangleq [x_{i,j,k,l}]_{i\in\mathbf{I},j\in\mathbf{J},k\in\mathbf{J},l\in\mathbf{K}}$ with elements $x_{i,j,k,l} = 1$ if task k is performed by executor i directly after task j and in mode l (0 if otherwise). Binary requirement is formally given as follows

$$\forall i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}, l \in \mathbf{K} \quad x_{i,j,k,l} \in \{0, 1\}. \tag{1}$$

Depending on the assignment, different amounts of resources are spent. As $e_{i,k,l}$ denoted is the cost at which the ith executor performs kth task in lth mode. The portion of the task k that is completed throughout this execution is denoted as $\eta_{i,k,l}$. Transitions between tasks cost $\mu_{i,j,k}$ for ith executor performing task k directly after task j. Those values are assumed non-negative, i.e. $e_{i,k,l} \geq 0, \eta_{i,k,l} \geq 0, \mu_{i,j,k} \geq 0$. A shorthand is defined as $e \triangleq [e_{i,k,l}]_{i\in\mathbf{I},k\in\mathbf{J},l\in\mathbf{K}}$, $\eta \triangleq [\eta_{i,k,l}]_{i\in\mathbf{I},k\in\mathbf{J},l\in\mathbf{K}}$, $\mu \triangleq [\mu_{i,j,k}]_{i\in\mathbf{I},j\in\mathbf{J},k\in\mathbf{J}}$. The constraints are divided into two sets of allocation-specific and routing-specific requirements.

First set ensures that at most one mode of execution is allowed for each executor-task pair, that task is completed when its completion rate of E has been achieved, the amount of resource F for each executor is limited:

$$\forall i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J} \quad \sum_{l\in\mathbf{K}} x_{i,j,k,l} \leq 1, \tag{2}$$

$$\forall k \in \mathbf{J}\backslash\{1\} \quad \sum_{i\in\mathbf{I},j\in\mathbf{J},l\in\mathbf{K}} \eta_{i,k,l} x_{i,j,k,l} \geq E, \tag{3}$$

$$\forall i \in \mathbf{I} \quad \sum_{j\in\mathbf{J},k\in\mathbf{J},l\in\mathbf{K}} (e_{i,k,l} + \mu_{i,j,k}) x_{i,j,k,l} \leq F. \tag{4}$$

It is assumed that $E > 0$ and $F > 0$. Furthermore, task $j = 1$ is treated as a depot (starting and ending location for all executors) therefore $\eta_{i,1,l} = e_{i,1,l} = 0$.

Second set ensures an uniform starting location for each executor, connectivity of the routes, lack of loops and lack of subcycles:

$$\forall i \in \mathbf{I} \quad \sum_{k \in \mathbf{J}, l \in \mathbf{K}} x_{i,1,k,l} = 1, \tag{5}$$

$$\forall i \in \mathbf{I}, k \in \mathbf{J} \quad \sum_{j \in \mathbf{J}, l \in \mathbf{K}} x_{i,j,k,l} = \sum_{j \in \mathbf{J}, l \in \mathbf{K}} x_{i,k,j,l}, \tag{6}$$

$$\forall i \in \mathbf{I}, k \in \mathbf{J} \quad \sum_{j \in \mathbf{J}, l \in \mathbf{K}} x_{i,j,k,l} \leq 1, \tag{7}$$

$$\forall i \in \mathbf{I}\, \forall S \subset \mathbf{J} \wedge S \neq \emptyset \wedge S \neq \mathbf{J} \quad \sum_{j \in S, k \in S, l \in \mathbf{K}} x_{i,j,k,l} \leq |S| - 1 + \frac{J}{J - |S|} - \frac{1}{J - |S|} \sum_{j \in \mathbf{J}, k \in \mathbf{J}, l \in \mathbf{K}} x_{i,j,k,l}. \tag{8}$$

These requirements do not prohibit some executors from avoiding selected tasks entirely but ensure that every executor visits every task at most once on their route.

Movement and task execution requires resource spendings and this generalized cost is encapsulated in the quality criterion given as follows

$$Q(x) \triangleq \sum_{i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}, l \in \mathbf{K}} x_{i,j,k,l}(e_{i,k,l} + \mu_{i,j,k}). \tag{9}$$

The main problem of this paper, which is simultaneous task allocation and routing, can be now defined.

Problem 1 (TAR – task allocation and routing).
Given: $\mathbf{I}, \mathbf{J}, \mathbf{K}, e, \eta, \mu, E, F$
Find: x^* where

$$x^* \triangleq \arg\min_{x \in \mathbf{D}} Q(x), \tag{10}$$

$$\mathbf{D} \triangleq \{x \in \mathbf{X} \triangleq \mathbf{R}_{i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}, l \in \mathbf{K}} : (1) \wedge \ldots \wedge (8)\}. \tag{11}$$

As was shown in [Ho1] this problem is NP-hard even in its feasibility version (in fact, just the allocation part is).

3 Solution Algorithm

Solution algorithm designed for TAR is based on the idea of a multi-stage functional decomposition. The original problem is divided into mult(-robot) task allocation MTA and mobile routing (MR) problems. The MTA itself is further

divided into several single-task relaxed allocation problems (STA) and relaxed task allocation problems (RTA). Individual problems and steps of the decomposition are given formally as follows. We recall the individual problems and algorithms after [Ho1,Ho2].

To define MTA, let the decision variable, which determines the allocation of tasks to executors, is denoted as $\breve{x} \triangleq [\breve{x}_{i,k,l}]_{i\in\mathbf{I},k\in\mathbf{J},l\in\mathbf{K}}$, where $\breve{x}_{i,k,l} = 1$ if executor i is assigned to task k in mode l (0 if otherwise).

*Problem 2 (*MTA – *multi-task allocation).*
Given: $\mathbf{I}, \mathbf{J}, \mathbf{K}, \eta, e, E, F$
Find: $\breve{x}^*$ where

$$\breve{x}^* \triangleq \arg\min_{\breve{x}\in\breve{\mathbf{D}}} \breve{Q}(\breve{x}), \quad \breve{Q}(\breve{x}) \triangleq \sum_{i\in\mathbf{I},k\in\mathbf{J},l\in\mathbf{K}} \breve{x}_{i,k,l} e_{i,k,l}, \tag{12}$$

$$\breve{\mathbf{D}} \triangleq \{\breve{x} \in \breve{\mathbf{X}} \triangleq \mathsf{X}_{i\in\mathbf{I},k\in\mathbf{J},l\in\mathbf{K}}\mathbf{R} : (14) \wedge \ldots \wedge (18)\}. \tag{13}$$

$$\forall i \in \mathbf{I}, k \in \mathbf{J}, l \in \mathbf{K} \quad \breve{x}_{i,k,l} \in \{0,1\}, \tag{14}$$

$$\forall i \in \mathbf{I}, k \in \mathbf{J} \quad \sum_{l\in\mathbf{K}} \breve{x}_{i,k,l} \leq 1, \tag{15}$$

$$\forall k \in \mathbf{J}\backslash\{1\} \quad \sum_{i\in\mathbf{I},l\in\mathbf{K}} \breve{x}_{i,k,l}\eta_{i,k,l} \geq E, \tag{16}$$

$$\forall i \in \mathbf{I} \quad \sum_{k\in\mathbf{J},l\in\mathbf{K}} \breve{x}_{i,k,l} e_{i,k,l} \leq F, \tag{17}$$

$$\forall i \in \mathbf{I}, l \in \mathbf{K} \quad \breve{x}_{i,1,l} = 1 \Leftrightarrow l = 1. \tag{18}$$

The RTA problem is formulated for a fixed task $k \in \mathbf{K}$. Let $\dot{x}_k \triangleq [\dot{x}_{i,k,l}]_{i\in\mathbf{I},l\in\mathbf{K}}$ be a decision variable (for a fixed $k \in \mathbf{J}$) with binary elements $\dot{x}_{i,k,l} = 1$ if task k is performed in lth mode by ith executor. The problem of a single task allocation is given as follows.

*Problem 3 (*STA – *single task allocation).*
Given: $d > 0, k \in \mathbf{J}, E, F, \mathbf{I}, \eta, e, \breve{x}$
Find: $\dot{x}_k^*$ where

$$\dot{x}_k^* \triangleq \arg\min_{\dot{x}_k\in\dot{\mathbf{D}}_k} \sum_{i\in\mathbf{I},l\in\mathbf{K}} \dot{x}_{i,k,l} e_{i,k,l} \tag{19}$$

$$\dot{\mathbf{D}}_k \triangleq \{\dot{x}_k \in \times_{i\in\mathbf{I},l\in\mathbf{K}}\mathbf{R} : (21) \wedge \ldots \wedge (25)\} \tag{20}$$

$$\forall i \in \mathbf{I}, l \in \mathbf{K} \quad \dot{x}_{i,k,l} \in \{0,1\}, \tag{21}$$

$$\forall i \in \mathbf{I} \quad \sum_{l\in\mathbf{K}} \dot{x}_{i,k,l} \leq 1, \tag{22}$$

$$\sum_{i\in\mathbf{I},l\in\mathbf{K}} \dot{x}_{i,k,l}\eta_{i,k,l} \geq E, \tag{23}$$

$$\forall i \in \mathbf{I} \quad \sum_{l\in\mathbf{K}} \dot{x}_{i,k,l} e_{i,k,l} \leq F, \tag{24}$$

$$\forall i \in \mathbf{I}, l \in \mathbf{K} \quad \dot{x}_{i,1,l} = 1 \Leftrightarrow l = 1. \tag{25}$$

Finally, we consider a relaxed task allocation problem RTA. The definition is as in STA with constraint (21) relaxed into

$$\forall i \in \mathbf{I}, l \in \mathbf{K} \quad \dot{x}_{i,k,l} \in [0,1], \tag{26}$$

and an additional constraint

$$\forall i \in \mathbf{I}, l \in \mathbf{K} \quad \dot{x}_{i,k,l} e_{i,k,l} \leq \min\{F, d\}. \tag{27}$$

where d is a positive parameter.

Since RTA is a linear programming problem it can be solved optimally with linear programming methods. Let us denote as $RTA(d)$ the solution to such a problem for parameter d. This solution can be rounded to be a 2-approximate solution of STA using a rounding algorithm RA. Let us also denote as $RA(d)$ the result of rounding with RA of $RTA(d)$.

Algorithm 1 RA

Let $i := 1, \dot{x} = [0]_{i \in \mathbf{I}, l \in \mathbf{K}}$
if exists only one $l \in \mathbf{K}$ for which $\dot{x}_{i,k,l} \in (0,1]$ **then**
    set $\dot{x}_{i,k,l} = 1$.
if exist two different $l_1, l_2 \in \mathbf{K}$ for which $\dot{x}_{i,k,l_1}, \dot{x}_{i,k,l_2} \in (0,1)$ and $\eta_{i,k,l_2} \geq \eta_{i,k,l_2}$ **then**
    set $\dot{x}_{i,k,l_1} = 1$, $\dot{x}_{i,k,l_2} = 0$.
return $\dot{x}$.

To solve the STA problem, algorithm A1 is used. Let us denote as $A1(k)$ the result of A1 for the kth task.

Algorithm 2 A1

Set $\dot{x}_k := [0]_{i \in \mathbf{I}, l \in \mathbf{K}}$, $j := 0$, $\overline{d}^j := \sum_{i \in \mathbf{I}} \max_{l \in \mathbf{K}} e_{i,k,l}$, $\underline{d}^j := \min_{i \in \mathbf{I}, l \in \mathbf{K}} e_{i,k,l}$.
Set $j := j + 1$.
Set $d^j := \lfloor (\overline{d}^{j-1} + \underline{d}^{j-1})/2 \rfloor$.
if $RTA(d^j)$ exists and if $\hat{Q}[RA(d^j)] \leq 2d$ **then**
    set $\dot{x} := RA(d^j)$,
    set $\overline{d}^j := d^j - 1$, $\underline{d}^j := \underline{d}^{j-1}$,
  else set $\underline{d}^j := d^j + 1$, $\overline{d}^j := \overline{d}^{j-1}$.
if $\underline{d}^j \leq \overline{d}^j$ **then**
    go to 2.
return $\dot{x}$.

For solving the MTA problem, algorithm A2f is used.

Next formulated is the mobile routing problem. Given a solution of MTA $\breve{x}$ let $\tilde{\mathbf{J}}_i \triangleq \{k \in \mathbf{J} : \sum_{l \in \mathbf{K}} \breve{x}_{i,k,l} = 1\}$ be a set of tasks to which the ith executor was

Algorithm 3 A2f

1: Set $k := 1$.
2: $\breve{x}_k \triangleq [\breve{x}_{i,k,l}]_{i\in\mathbf{I},l\in\mathbf{K}} := A1(k)$.
3: If $\breve{x}_k$ does not exits then **return** no solution.
4: $F_i := F_i - \sum_{i\in\mathbf{I},l\in\mathbf{K}} \breve{x}_{i,k,l} e_{i,k,l}$.
5: $k := k + 1$.
6: If $k \leq J$ go to 2. **return** $\breve{x} \triangleq [\breve{x}_{i,k,l}]_{i\in\mathbf{I},k\in\mathbf{J},l\in\mathbf{K}}$.

assigned. Let us denote the route of the ith executor as $y_i \triangleq [y_{i,j,k}]_{j\in\tilde{\mathbf{J}},k\in\tilde{\mathbf{J}}_i}$ where $y_{i,j,k} = 1$ if task k is executed directly after task j (0 if otherwise). Problem MR is given as follows.

Problem 4 (MR – mobile routing).
Given: $i \in \mathbf{I}, \tilde{\mathbf{J}}_i, \mu$ Find: y_i^* where

$$y_i^* \triangleq \arg\min_{y_i\in\mathbf{D}_{\mathrm{y}}^i} Q_{\mathrm{y}}^i(y_i), \quad Q_{\mathrm{y}}^i(y_i) \triangleq \sum_{j\in\tilde{\mathbf{J}}_i,k\in\tilde{\mathbf{J}}_i} y_{i,j,k}\mu_{i,j,k} \tag{28}$$

$$\mathbf{D}_{\mathrm{y}}^i \triangleq \{y_i \in \mathsf{X}_{j\in\tilde{\mathbf{J}}_i,k\in\tilde{\mathbf{J}}_i}\mathbf{R} : (30)\wedge(31)\wedge(32)\wedge(33)\}. \tag{29}$$

$$\forall j \in \tilde{\mathbf{J}}_i, k \in \tilde{\mathbf{J}}_i \quad y_{i,j,k} \in \{0,1\}, \tag{30}$$

$$\forall j \in \tilde{\mathbf{J}}_i \quad \sum_{k\in\tilde{\mathbf{J}}_i} y_{i,1,k} = 1, \tag{31}$$

$$\forall j \in \tilde{\mathbf{J}}_i \quad \sum_{j\in\tilde{\mathbf{J}}_i} y_{i,j,k} = \sum_{j\in\tilde{\mathbf{J}}_i} y_{i,k,j}, \tag{32}$$

$$\forall S \subset \tilde{\mathbf{J}}_i \wedge S \neq \emptyset \wedge S \neq \tilde{\mathbf{J}}_i \quad \sum_{j\in S,k\in S'} y_{i,j,k} \geq 2. \tag{33}$$

MR is in fact an instance of the Travelling Salesman Problem and can therefore be solved with dedicated methods. We will use the classic cheapest insertion algorithm. Given solutions $\breve{x}$ and y we can obtain a solution to TAR using the following equation

$$x_{i,j,k,l} \triangleq \breve{y}_{i,j,k}\breve{x}_{i,k,l}. \tag{34}$$

Finally, presented is the solution algorithm to the TAR problem. It is given as follows.

Algorithm 4 TARsol

1: Formulate and solve SMTA to obtain $\breve{x}$.
2: Formulate and solve MR under $\breve{x}$. Obtain y.
3: Use (34) to obtain x.

This dedicated method of solving TAR is compared empirically with algorithms for the Multidimensional Knapsack and Covering problem in the following section.

4 Empirical evaluation

In this section we compare the TARsol with the Adaptive Memory Search (AMS) [AHL] as well as the Alternating Control Tree (ACT) [HALG]. Both methods are of the iterative improvement type and were tested for varying limits on the maximum number of iterations. Since both methods provide means of solving a maximization problem, the objective function had to be negated. Additionally, for Alternating Control Tree, the constraint (4) in [HALG] had to be modified accordingly. For solving linear programming problems and integer programming problems a solver *GLPK* [glpk] was used.

We use the following abbreviations for tested algorithms:

- $AMS - x$ which is the Adaptive Memory Search method run for up to x iterations,
- $ACT - x$ which is the Alternating Control Tree method run for up to x iterations.

Algorithm AMS requires determination of several parameters. The evaluation of their influence on the solution is presented in Series 1 of the experiments.

Series 1.

Tested were the following parameters of the AMS algorithm (with default values): number of iterations N, tabu tenure($base = 10$) and the weight update coefficients $\alpha_* = 1, \alpha_+ = 0, \beta_* = 1, \beta_+ = 0, w_{inc} = 0$. The experiment was done for the problem data: $I = 3$, $J = 4$, $L = 2$, $\mu_{i,1,j} = \mu_{i,j,1} = 1$, for $j, k \geq 2$: $\eta_{i,k,l} = l$, $e_{i,k,l} = 10 + l$, $\mu_{i,j,k} = |j - k|$, $F_i = 50$, $E = 2$.

Due to the random nature of AMS algorithm, every experiment was repeated 10 times and the results were averaged over all runs. They are presented in the corresponding tables where *Succ* is the number of experiments where a solution was found, Q is the average quality and T is the average execution time (in seconds).

Experiment 1.1 Tested is the number of iterations N. Results are presented in Table 1. For further tuning selected was the value of $N = 1000$.

N	10	20	50	100	200	500	1000	2000	5000
Succ	0	0	4	3	2	4	5	5	4
Q	-	-	108.25	82.33	84	106.75	85.6	90.2	69.5
T	-	-	0.56	1.06	2.06	5.09	10.04	20.22	38.09

Table 1. Results of Experiment 1.1. *Succ*, Q and T for varying N.

Experiment 1.2 Tested is the tabu tenure *base*. Results are presented in Table 2. For further testing, selected was the value of $base = 60$.

Experiment 1.3 Tested are the additive weighting coefficients $\alpha_+ = \beta_+$. Results are presented in Table 3. Selected were the values of $\alpha_+ = \beta_+ = 0$.

Experiment 1.4 Tested are the multiplicative weighting coefficients $\alpha_* = \beta_*$. Results are presented in Table 4. Selected were the values of $\alpha_* = \beta_* = 1$.

base	1	2	5	10	20	50	60	70	80
Succ	5	3	3	6	8	10	10	7	8
Q	87.2	75.67	103.33	80.67	89.75	90.1	84.7	80.43	88.86
T	9.67	9.67	9.88	10.23	10.2	12.76	13.99	13.56	14.59

Table 2. Results of Experiment 1.2. *Succ*, Q and T for varying *base*.

base	0	0.1	0.2	0.5	1	2	5
Succ	10	9	10	10	10	10	9
Q	80	81,67	81,7	81,6	85,3	84,7	85,11
T	12.8	12.74	12.9	13.17	13.46	14.19	14.71

Table 3. Results of Experiment 1.3. *Succ*, Q and T for varying α_+, β_+.

Experiment 1.5 Tested are the weighting coefficient w_{inc}. Results are presented in Table 5. Selected was the value value of $w_{inc} = 0$.

It is clear from the experiments that that basic version of AMS barely manages to solve TAR even for a very simple instance. Increasing the tabu tenure resulted in a significant increase in number of successful experiments without a major change in the average execution time. It is also clear from Tables 3-5 that on-line modifications to weighting coefficients do not improve the results. Those parameters were kept at their default values.

Series 2.
In this series compare are TARsol, ACT and AMS for a selected instance of the TAR problem. Tested is the dependence of quality of the solution and the execution time on the number of tasks. Results are presented in Tables 6 and 7. The execution of AMS was repeated 5 times for each set of parameters and the results are the average. The number of successful executions is provided in brackets or not at all if every execution succeeded to provide a feasible solution. Parameters of the problem are as follows: $I = 3$, $J \in \{4, 5, 6\}$, $L = 2$, $\mu_{i,1,l} = \mu_{i,l,1} = 1$, for $j, k \geq 2 : \eta_{i,k,l} = l$, $e_{i,k,l} = 10 + l$, $\mu_{i,j,k} = |j - k|$, $F_i = 100$, $E = 2$.

base	1	1.1	1.2	1.5	2
Succ	10	6	6	6	3
Q	85.4	74.67	96	91.33	81.67
T	13.77	11.94	12.11	11.79	11.82

Table 4. Results of Experiment 1.4. *Succ*, Q and T for varying α_*, β_*.

base	0	0.1	0.2	0.5	1
Succ	10	3	0	0	0
Q	90.2	78	-	-	-
T	13.96	12.94	-	-	-

Table 5. Results of Experiment 1.5. *Succ*, Q and T for varying w_{inc}.

J	*TARsol*	$ACT-200$	$AMS-1000$	$ACT-500$	$AMS-2000$
4	42	62	88.25 (4)	62	84.8
5	59	(0)	141 (1)	77	116.5 (4)
6	77	(0)	(0)	93	173 (1)

Table 6. Results of Series 2. Q for varying J.

J	*TARsol*	$ACT-200$	$AMS-1000$	$ACT-500$	$AMS-2000$
4	0.06	1.91	12.59 (4)	21.67	28.28
5	0.07	(0)	29.29 (1)	53.07	40.21 (4)
6	0.09	(0)	(0)	105.83	79.6 (1)

Table 7. Results of Series 2. T for varying J.

Concluding results of Series 2, we can observe a clear advantage of TARsol over AMS and ACT for the tested cases. Both the quality of obtained solutions and the execution time is better for TARsol, the latter by several orders of magnitude. The difference between AMS and ACT alone is similarly straightforward, ACT provides better solutions in a shorter span of time. Furthermore, for every tested case, the TARsol alone provided a feasible solution every time. Obtaining feasibility proved to be most difficult for the tested versions of the AMS algorithm where it often failed, even for small instances.

The main cause behind the observed results seems to lie in the size of the TAR problem. The number of variables and constraints is overwhelming to tackle all at once. Even ACT, which performs a decomposition on its own, did not perform comparably to TARsol.

5 Conclusion

The complexity of TAR prohibits a direct application of the AMS and ACT methods. Both, the execution time and the quality of the solution obtained by the dedicated algorithm TARsol are, on average, better. Furthermore, the algorithm tends to find a solution for cases when AMS and ACT fail. Such results are consistent with those obtained for the simpler MTA which was tackled in [Ho2]. Further study on the properties of TARsol and on other dedicated solution methods is necessary. Use of decomposition in conjunction with AMS and ACT should also be evaluated.

References

[AHL] Arntzen, H., Hvattum L., Lokketangen A.: Adaptive memory search for multi-demand multidimensional knapsack problems. Computers & Operations Research

33, pp. 2508–2525. Elsevier (2005)
[CV] Coltin B., Veloso M., Mobile Robot Task Allocation in Hybrid Wireless Sensor Networks, *Proceedings of International Conference on Intelligent Robots and Systems*, pp. 2932–2937 (2010)
[CM] Correll N., Martinoli A., Multirobot Inspection of Industrial Machinery, *IEEE Robotics and Automation Magazine*, Vol. 16, pp. 103–112 (2009)
[Ho1] Hojda M., Task allocation in robot systems with multi-modal capabilities, *IFAC-PapersOnLine, 15th IFAC Symposium on Information Control Problems in Manufacturing – INCOM 2015* Vol. 48, No. 3, pp. 2109-2114, Elsevier (2015)
[Ho2] Hojda M., Comparison of algorithms for constrained multi-robot task allocation, *Advances in Systems Science : proceedings of the International Conference on Systems Science*, pp. 255-264, Springer (2017)
[HALG] Hvattum L., Arntzen, H., Lokketangen A., Glover F.: Alternating control tree search for knapsack/covering problems. Journal of Heuristics 16, pp. 239–258. Springer (2008)
[JS] Jung D., Savvides A., An Energy Efficiency Evaluation for Sensor Nodes with Multiple Processors, Radios and Sensor, *Proceedings of the 27th IEEE Conference on Computer Communications*, pp. 1112–1120 (2008)
[LLFNS] Li X., Lille I., Falcon R., Nayak A., Stojmenovic I., Servicing Wireless Sensor Networks by Mobile Robots, *IEEE Communications Magazine*, Vol. 50, No. 7, pp. 147–154 (2012)
[MPA] Melodia Tl, Pompili D., Akyildiz I., Handling Mobility in Wireless Sensor and Actor Networks, *IEEE Transactions on Mobile Computing*, Vol. 10, No. 2, pp. 160–173 (2010)
[RSSH] Rahimi M., Shah H., Sukhatme G., Heideman J., Studying the Feasibility of Energy Harvesting in a Mobile Sensor Network *Proceedings of ICRA '03. IEEE International Conference on Robotics and Automation, 2003*, Vol. 1, pp. 14–19 (2003)
[SBCFTW] Shott B., Bajura M., Czarnaski J., Flidr J., Tho T., Wang L., A modular power-aware microsensor with 1000x dynamic power range, *Proceedings of Information Processing in Sensor Networks*, pp. 469–474 (2005)
[TI] Tekdas O., Isler V., Using Mobile Robots to Harvest Data from Sensor Fields, *IEEE Wireless Communications*, Vol. 16, No. 1, pp. 22–28 (2009)
[TLLB] Tirta Y., Li Z., Lu Y-H., Bagchi S., Efficient Collection of Sensor Data in Remote Fields Using Mobile Collectors, *Proceedings of 13th International Conference on Computer Communications and Networks*, Vol. 50, No. 7, pp. 147–154 (2012)
[YMHC] Yong F., Mo S., Hackmann G., Chenyang L., Practical control of transmission power for Wireless Sensor Networks, *Proceedings of 20th IEEE International Conference on Network Protocols*, pp. 1–10 (2012)
[glpk] GNU Linear Programming Kit, https://www.gnu.org/software/glpk/