Rotational Diversity in Multi-Cycle Assignment Problems

Helge Spieker, Arnaud Gotlieb
Simula Research Laboratory
P.O. Box 134
1325 Lysaker, Norway
{helge,arnaud}@simula.no

Morten Mossige
University of Stavanger
Stavanger, Norway
ABB Robotics
Bryne, Norway
morten.mossige@uis.no

Abstract

In multi-cycle assignment problems with rotational diversity, a set of tasks has to be repeatedly assigned to a set of agents. Over multiple cycles, the goal is to achieve a high diversity of assignments from tasks to agents. At the same time, the assignments' profit has to be maximized in each cycle. Due to changing availability of tasks and agents, planning ahead is infeasible and each cycle is an independent assignment problem but influenced by previous choices. We approach the multi-cycle assignment problem as a two-part problem: Profit maximization and rotation are combined into one objective value, and then solved as a General Assignment Problem. Rotational diversity is maintained with a single execution of the costly assignment model. Our simple, yet effective method is applicable to different domains and applications. Experiments show the applicability on a multi-cycle variant of the multiple knapsack problem and a real-world case study on the test case selection and assignment problem, an example from the software engineering domain, where test cases have to be distributed over compatible test machines.

1 Introduction

In this paper, we address the problem where multiple cycles of assignment problems have to be solved, with the additional goal to assign each task to different agents in subsequent cycles and eventually to all compatible agents.

In general, assignment problems are well-studied in artificial intelligence and can be solved efficiently. Their goal is to assign a set of weighted tasks to a set of agents, such that capacity constraints are satisfied and a profit function is maximized. These problems are relevant in a broad context, of which many consider rotation aspects, too. In nurse rostering (Chiaramonte 2008; Azizi, Zolfaghari, and Liang 2010) and workforce scheduling (Ernst et al. 2004; Musliu, Schutt, and Stuckey 2018), rotation is relevant to avoid boredom and fatigue or to cover a constrained shift system. In aircraft rotation (Clarke et al. 1996) or machine scheduling (Ma, Chu, and Zuo 2010), rotation mechanisms allow to keep maintenance schedules or optimize usage patterns of machinery. Not always can such scheduling requirements and needs be addressed upfront, albeit for changing demand patterns, personnel availability due to vacations and sickness,

or short-term planning horizons for other reasons. Therefore, it is necessary to include rotation mechanisms for iterative, recurring planning scenarios.

The presented approach addresses multi-cycle assignment problems with varying availability of tasks and agents under the additional goal to rotate assignments from tasks to agents over multiple cycles. Tasks and agents can be unavailable for one or several cycles without previous notice or information about their next availability. We refer to the subsequent diverse assignments as *rotational diversity*. A full example of the problem and our solution is given in Section 4.1.

This work defines the term *rotational diversity* in a temporal manner. That is, the solution to multiple subsequent instances of a problem has to differ in the assignments made. This is different from the notion of solution diversity, where it is desirable to find multiple distinctly diverse solutions to one instance of a problem (Glover, Løkketangen, and Woodruff 2000; Hebrard et al. 2005; Trapp and Konrad 2015; Petit and Trapp 2015).

We present an optimization model that combines profits and affinities, a metric to describe the state of rotation, into a single optimization criterion. Solving this model incrementally, that is, at each cycle, allows to control rotational diversity. A central component for this control is the strategy, that defines how profits and affinities are combined.

We discuss five strategies for this combination of values and evaluate them on two case studies. A first case study is a multi-cycle extension of the multiple knapsack problem, and the second is a real-world case study of test case selection and assignment problem (TCSA), originating from the problem of testing software for robotic systems. Our results show that rotation is achieved while sacrificing less than 4 % of the original goal of profit maximization in TCSA.

2 Related Work

The general multi-cycle assignment problem is a variant of the *General Assignment Problem* (GAP) (Pentico 2007). A set of tasks, each associated with a profit and a weight, has to be assigned to a set of agents with limited capacity. The goal is to maximize (or minimize) the summed profits of the assigned tasks, while the weights do not exceed the agent capacities. Not all tasks have to be assigned, and profits and weights can vary between agents. The classical assignment problem formulates a cost minimization objective, although

maximization, which we use throughout this work, is also commonly found in problem variants.

In this paper, we formulate rotational diversity as a general assignment problem, as our contribution is steered towards the general rotation mechanism. The closest problem variant is the group of knapsack problems. One or multiple agents have to be filled to maximize the value of the selected tasks (Martello and Toth 1987). A multi-cycle knapsack variant is presented in (Faaland 1981), although only the unassigned items from previous cycles are available in subsequent cycles.

Assignment rotation is found in job rotation scheduling. Here, a common goal is to find schedules and work assignments for humans to avoid fatigue and boredom (Bhadury and Radovilsky 2006), or to evenly distribute shifts to personnel (Bard and Purnomo 2005; Ayough, Zandieh, and Farsijani 2012). This is approached by a static schedule, where the assignment between workers and their tasks frequently changes. We do not fix one assignment over multiple cycles, but have to repeatedly create individual assignments at each cycle due to changing availability of agents and tasks. Opposite to diverse rotations is the concept of *persistence* in robust optimization (Bertsimas, Natarajan, and Teo 2006; Morrison 2010). Persistence (Brown, Dell, and Wood 1997; Bertsimas, Natarajan, and Teo 2006) considers finding stable assignments during optimization, such that improvements in the solution objective only cause small changes in the variable assignment of the solution. By maximizing the affinity between tasks and agents, we can adjust the presented method to support persistent instead of diverse assignments in subsequent iterations of a problem using similar techniques. We note, that the concept of affinity, which we introduce in Section 3.1, can be transferred to multi-cycle problems with persistence as well.

Fair allocations, which maximize a social welfare function, are considered in game theory research. Mechanisms for the resource distribution include *combinatorial auctions* and *exchanges* (de Vries and Vohra 2003; Endriss et al. 2006). Both have shown to result in a balanced and fair distribution of resources, although it is complex to determine which resources to offer in an auction or exchange and who is the resulting winner (Sandholm and Suri 2003). Recent works further discuss aspects of repeated matching between tasks and agents, under consideration of dynamic preferences and fairness (Hosseini, Larson, and Cohen 2015), or repeated matching of previously unmatched tasks (Anshelevich et al. 2013). Because combinatorial auctions and exchanges can be decentralized, these techniques are commonly used for resource allocation in multi-agent systems (Liu and Mohamed 2008; Nongaillard et al. 2009).

In this work, we do not directly solve the GAP, but instrument a general solver to maintain a fair distribution of tasks to agents. An alternative is a system where agents exchange tasks among them to achieve rotation. However, preliminary experiments showed this approach to be inferior to the one presented. The evaluated exchange model first focused on profits only, and afterwards aimed for a fair rotation by allowing one-task exchanges between agents. It showed that a good GAP solution leaves only few choices for one-task
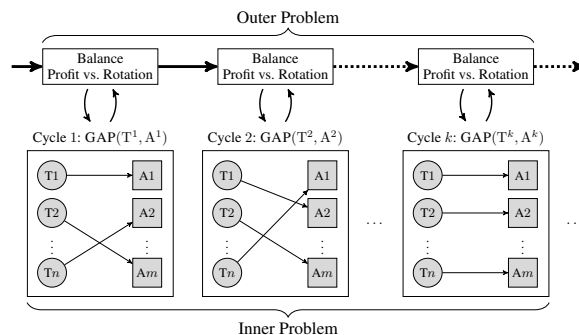

Figure 1: Multi-Cycle Assignment Problem: At each cycle an independent GAP has to be solved.

exchanges and only minimal improvements in rotational diversity occurred.

3 Problem Description

We first introduce the multi-cycle assignment problem as a combination of two problems, which we further define afterwards. We discuss the characteristics of the class of general assignment problems at the core of our approach. Finally, we formulate and discuss the requirements for maintaining multi-cycle rotational diversity.

In multi-cycle assignment problems, every cycle is a distinct planning unit, because, due to the availability of tasks and agents, planning ahead is not possible. Therefore, rotational diversity has to be considered at every cycle.

This separates the overall problem into two partial problems (as visualized in Figure 1): First, the *inner problem* is to solve an independent GAP in each cycle k. The GAP selects a subset of the available tasks while maximizing the sum of their values.

Second, the *outer problem* aims to maintain a diverse assignment between tasks to agents, meaning that the tasks are frequently assigned to all compatible agents over subsequent cycles. As a mechanism for this balance, we utilize the *affinity* between a single task and each agent, and the *affinity pressure* as a metric to evaluate the whole set of tasks and agents. The balancing mechanism between profit optimization and rotation of tasks, is called a *strategy*.

The inner problem, as well as both the affinity and the affinity pressure will be further defined and introduced in the following section. As part of our method, we introduce five strategies for achieving rotational diversity in Section 4.2.

3.1 Multi-Cycle General Assignment Problem

The general assignment problem, $\text{GAP}(\mathcal{T}^k, \mathcal{A}^k)$, receives as inputs the tasks and agents available at cycle k. The set of agents, $\mathcal{A}^k$, consists of m integers i, each with a fixed *capacity*, b_i, and the set of tasks, $\mathcal{T}^k$, consists of n integers j. Both sets are given at each cycle and can unpredictably change from cycle k to $k + 1$.

The relation between a task and an agent has three fixed attributes: both the *profit* p_{ij} and the *weight* w_{ij} are externally fixed and describe the benefit respectively the resource

demand of task j when assigned to agent i. Each task further has a set of *compatible agents*, $\mathcal{C}_j^k$, that it can be assigned to.

The *affinity* a_{ij} is not fixed, but changes between cycles. The affinity numerically describes the preferred assignments from tasks to agents, with higher values giving a higher preference for a task to be assigned to that agent.

Additionally, we refer to *values* in the context of the optimization objective of the GAP. Here, the value v_{ij} is a combination of profits and affinities, a way to balance profit- and rotation-oriented assignments. For a standard assignment problem without affinities, the values equal the profits.

The affinity between a task and an agent, a_{ij}, is the number of cycles since the last assignment of task j to agent i. The affinity quantifies the preference of a task to be assigned to certain agents during the next cycles. The affinity pressure is the maximum of all affinities in the set of tasks. Both the affinity and the affinity pressure will be further discussed after a definition of the inner assignment problem.

Definition 1. Multi-Cycle General Assignment Problem

$$\text{Maximize} \sum_{i \in \mathcal{A}^k} \sum_{j \in \mathcal{T}^k} x_{ij} v_{ij} \tag{1}$$

$$\text{subject to} \sum_{j \in \mathcal{T}^k} x_{ij} w_{ij} \leq b_i, \qquad \forall i \in \mathcal{A}^k \tag{2}$$

$$\sum_{i \in \mathcal{A}^k} x_{ij} \leq 1, \qquad \forall j \in \mathcal{T}^k \tag{3}$$

with

k : Index of the current cycle

$\mathcal{A}^k$: A set of integers i labeling m agents

$\mathcal{T}^k$: A set of integers j labeling n tasks

b_i : Capacity of agent i

v_{ij} : Value of task j when assigned to agent i $\qquad$ (4)

w_{ij} : Weight of task j on agent i

$$x_{ij} : \begin{cases} 1 & \text{Task } j \text{ is assigned to agent } i \wedge i \in \mathcal{C}_j^k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The problem's objective is to maximize the total sum values of the assigned tasks (Eq. 1). Each agent can hold multiple tasks up to its resource limit (Eq. 2) and each task is assigned to at most one agent (Eq. 3). The assignment of tasks to agents is constrained by compatibility constraints (Eq. 5), such that each task can only be placed on a subset of agents.

We state a very general GAP formulation, although our proposed approach is able to handle different GAP variants. The most important and required properties of the formulation are: a) the possibility to have different values per task and agent (Eq. 4), and b) the value maximization objective (Eq. 1).

GAP is NP-hard as it reduces to the NP-hard one-dimensional knapsack optimization problem (Karp 1972).

3.2 Rotational Diversity

To maintain rotational diversity, it is necessary to control the affinities between tasks and agents. As an indicator, the affinity pressure must not grow too high, which can be avoided by a diverse rotation between tasks and agents.

Figure 2: In the outer problem, profits p and affinities a are combined by a strategy $\bigcirc$ into one values v. These values are used to optimize the GAP in the inner problem.

As part of solving the outer problem, it is necessary to balance profit maximization and reducing affinities by rotating the assignments from tasks to agents. Additional complexity stems from the fact, that at each cycle different sets of agents and tasks are available and the assignment can only take the current cycle into account.

The optimization in the outer problem could be solved by an exhaustive search of possible combinations between profits and affinities, such that an optimal solution can be found. In practice, this is infeasible, as it requires to solve the computationally expensive inner GAP problem multiple times before deciding for the final solution.

4 Solution Approach

The central idea to maintain rotational diversity is to manipulate the values contributing to the objective of the inner assignment problem (see Figure 2). This adjustment steers the optimization process towards an assignment which is balancing profit maximization and making diverse assignments. The adjustment is made according to a strategy and the state of the available resources, that is tasks and agents available in the current cycle, and their affinities.

Before introducing different adjustment strategies, we describe the mechanism to calculate the affinities and the affinity pressure, and the relevance of their values.

4.1 Assignment Diversity

To achieve rotation of tasks over agents in subsequent cycles, the cycle-specific assignment problem needs an incentive to assign a task to a different agent than in previous assignments.

This incentive is described by the notion of affinities between tasks and agents, describing how important an assignment of a task to an agent is to achieve high rotational diversity. A low affinity value corresponds to a recent assignment from the task to the agent, whereas a high affinity indicates the necessity to make this assignment again soon.

The affinities are determined by *Affinity Counting*.

Definition 2. Affinity Counting

$$a_{ij}^k = \begin{cases} 0 & \text{if } i \notin \mathcal{C}_j^k & \text{(6a)} \\ 1 & \text{if } k = 1 \vee x_{ij}^{k-1} = 1 & \text{(6b)} \\ a_{ij}^{k-1} + 1 & \text{if } i \in \mathcal{A}^k \wedge j \in \mathcal{T}^k & \text{(6c)} \\ a_{ij}^{k-1} & \text{otherwise} & \text{(6d)} \end{cases}$$

Affinity Counting counts the number of cycles since the last assignment from task j to agent i, starting from 1 at the

first cycle or the last assignment (6b). If a task and agent are incompatible, the affinity is always 0 (6a). At cycle k, the affinity increases for non-selected, but possible assignments in the previous cycle $k-1$ (6c)(6d).

Naturally, the affinity values increase over time as each task can only be assigned to one of the compatible agents in each cycle. This growth is anticipated and acceptable to a certain degree, while at the same time, growing affinities show the need to make the corresponding assignment soon.

To monitor the overall state of rotational diversity, we define the *Affinity Pressure* metric.

Definition 3. Affinity Pressure (AP)
The Affinity Pressure is defined per cycle k and task j:

$$\text{AP}_j^k = \frac{\sum_{i \in \mathcal{A}^k} a_{ij}^k}{|\mathcal{C}_j^k|} - \frac{|\mathcal{C}_j^k| + 1}{2}$$

It is the scaled difference between the actual and ideal affinities, as described below. For the AP calculation, only the task and agents available in that cycle are considered. Hence, tasks and agents can be added or completely removed without affecting the AP values of the remaining tasks.

In an ideal rotation setting, the affinities of a task j form the set $\{ i \mid 1 \le i \le |\mathcal{C}_j^k| \}$, with its sum being the triangular number $\frac{1}{2} \cdot |\mathcal{C}_j^k| \cdot (|\mathcal{C}_j^k| + 1)$. As the task is (ideally) assigned in every cycle, the last assignment has affinity 1, the previous assignment has affinity 2, and so on. With $|\mathcal{C}_j^k|$ compatible agents, the longest unassigned task then has affinity $|\mathcal{C}_j^k|$.

However, in a practical rotation setting, this perfect rotation is hindered by non-availability and limited capacities of the agents. To evaluate the state of rotational diversity, it is, therefore, crucial to consider how long a task has not been assigned to each agent, but also, from an agent's perspective, the time it has not executed certain tasks.

The AP metric is derived from the difference between the sum of current affinities and the ideal values. For comparability and normalization, it is scaled by the number of possible agents: $\frac{1}{|\mathcal{C}_j^k|} \cdot \left[\sum_{i \in \mathcal{A}^k} a_{ij}^k - \frac{1}{2} \cdot |\mathcal{C}_j^k| \cdot (|\mathcal{C}_j^k| + 1) \right]$

In this formula, the minuend describes the current affinities relative to the number of possible agents, the subtrahend the ideal case with fully regular rotation. A positive excess indicates missed assignments to achieve ideal rotation. Note that the bottom value of 0 is an ideal value, which in practice is usually not achievable, due to selection and limited availability of tasks and agents, and the necessary selection in the GAP assignment problem. During the first $|\mathcal{C}_j^k|$ cycles, the AP for a task is negative, as initially all affinities equal 1. After $|\mathcal{C}_j^k|$ cycles, the AP is always ≥ 0.

Example. Figure 3 presents an example of affinities and their development over four cycles. In the initial cycle 1, all affinities equal 1 (or 0 for incompatible assignments) and there is no preferred assignment among all possible assignments.

Over the next cycles, tasks T1 and T2 rotate over all compatible agents, resulting in the AP value 0 for T1 and T3. Task T3 does not rotate, but is assigned to agent C in two subsequent cycles, which increases the affinity for the assignment to agent B and raises the AP to 0.5, an indicator for the

	A	B	C	AP
T1	1	1	0	-0.5
T2	1	1	1	-1.0
T3	0	1	1	-0.5

(a) Cycle 1

	A	B	C	AP
T1	1	2	0	0
T2	2	1	2	-0.3
T3	0	2	1	0

(b) Cycle 2

	A	B	C	AP
T1	2	1	0	0
T2	1	2	3	0
~~T3~~	~~0~~	~~3~~	~~1~~	~~0.5~~

(c) Cycle 3

	A	B	C	AP
T1	1	2	0	0
T2	2	3	1	0
T3	0	3	1	0.5

(d) Cycle 4

Figure 3: Affinities and Affinity Pressure of three tasks T1, T2, T3 and agents A, B, C over four cycles (Bold: Ideal; Highlighted: Assignment in cycle k; Strikethrough: Task unavailable)

imbalance of T3. Note that, in cycle 3, T3 is unavailable, but this does not affect its affinities in cycle 4.

Theorem 1. For any set of tasks $\mathcal{T}^k$ and agents $\mathcal{A}^k$ with constant availability, if a task j is always assigned to one of the agents for which it has the highest affinity, a perfect rotation is achieved and the Affinity Pressure is 0.

Proof. With N possible agents, it takes N cycles to assign a task once to every agent. The affinity is set to 1 after the assignment was made and is increased by 1 at every cycle. After each assignment was made once, the affinity to the first assigned agent is N again, the affinity of the second assigned agent is $N-1$, and the affinity of the last assigned agent is 1. The sum of affinities is $\sum_{i \in \mathcal{A}^k} a_{ij}^k = \sum_{i=1}^{N} i = \frac{1}{2}N(N+1)$. Using Definition 3, and because the number of available agents is constantly $|\mathcal{C}_j^k| = N$, it follows $\text{AP} = 0$. $\qquad\square$

4.2 Strategies

A central strategy balances profit maximization and diverse assignments, by controlling the combination of profits and affinities into values. This combination then steers the focus of the single-objective GAP solver.

The general optimization scheme is such that, first, the state of the system, given by available tasks and agents, and the affinity pressure are gathered. Second, the task values are derived, and the cycle's GAP is solved. Finally, based on the actual assignments, the affinities of the available tasks are updated. This procedure adds little overhead to a process where no rotation is considered, as the main computational effort remains in the central GAP.

A strategy is fixed in the optimization process, such that it is not exchanged between cycles. However, a strategy can be adaptive towards the current state of tasks, agents, and affinities. At the beginning of every cycle, the strategy calculates the profit values, based on profits, affinities, and (if required) other information about the current state. These values are then taken as parameters in the current cycle's GAP instance.

In the following, we present five strategies to control rotational diversity and combine profits and affinities.

Strategy 1: Objective Switch (OS/γ)

The Objective Switch strategy maintains rotational diversity by monitoring the affinity pressure, and, if it reaches a threshold γ, switches from profit to affinity values:

$$v_{ij} \triangleq \begin{cases} p_{ij} & \text{if } \gamma > \max_{j \in \mathcal{T}^k} \text{AP}_j^k \\ a_{ij} & \text{otherwise} \end{cases}$$

The threshold γ is a fixed, user-defined configuration parameter, and selected according to the desired trade-off between maximized profits and high rotational diversity.

Strategy 2: Product Combination (PC)

In the Product Combination strategy, profit and affinities are multiplied to form the task values:

$$v_{ij} \triangleq p_{ij}^{\alpha} \cdot a_{ij}^{\beta}$$

The exponents α and β allow configuration of the strategy to emphasize one aspect or to account for different scales of profits and affinities in specific applications. However, we found using a standard configuration of $\alpha = \beta = 1$ to be intuitive and well-performing. Therefore, this strategy does not require additional configuration, but allows for adjustments if necessary.

In the PC strategy, there is not active reaction on the overall state of rotational diversity, as in the OS/γ strategy, but higher affinities values implicitly influence the profits and put an emphasis on tasks with missing rotation.

Strategy 3: Weighted Partial Profits (WPP)

The WPP strategy calculates task values with a weighted sum:

$$v_{ij} \triangleq \lambda_j^k \cdot \frac{p_{ij}}{\max\limits_{i \in \mathcal{A}^k} \max\limits_{j \in \mathcal{T}^k} p_{ij}} + (1 - \lambda_j^k) \cdot \frac{a_{ij}}{\max\limits_{i \in \mathcal{A}^k} \max\limits_{j \in \mathcal{T}^k} a_{ij}}$$

The task- and cycle-specific weight parameter λ_j^k balances the influence of each objective on the final value v_{ij}. λ_j^k is self-adaptive and depends on the ratio between ideal and actual affinities, similar to the affinity pressure. When the rotational diversity is high, the influence of the profits is high, too, otherwise the affinities have higher influence:

$$\lambda_j^k = \frac{\frac{1}{2} \cdot |\mathcal{C}_j^k| \cdot (|\mathcal{C}_j^k| + 1)}{\sum_{i \in \mathcal{A}^k} a_{ij}}$$

To account for different value ranges, both profits and affinities are scaled to $[0, 1]$ by their respective maxima.

Strategy 4: Fixed Objective: Profit (FOP)

Each task value equals the static profit value: $v_{ij} \triangleq p_{ij}$

Strategy 5: Fixed Objective: Affinity (FOA)

Each task value equals the affinity value: $v_{ij} \triangleq a_{ij}$

FOP and FOA represent special cases of the PC strategy, with $\beta = 0$ respectively $\alpha = 0$. These strategies are the two most extreme approaches, because each of them ignores the other goal, albeit profits or affinities. They serve as comparison baselines to evaluate the trade-offs by the other strategies.

5 Experimental Evaluation

We consider two problem types for evaluation: a) a multi-cycle variant (MCMKP) of the known multiple knapsack problem (MKP) to evaluate trade-offs between the strategies; b) test case selection and assignment (TCSA) as a real-world case study from the area of software testing to evaluate the practical interest of our approach.

5.1 Implementation and Setup

Our strategies and the experimental setup are implemented in Python. The assignment problem is modeled with MiniZinc 2.0 (Nethercote et al. 2007), following the presented GAP formulation, and is solved with IBM CPLEX 12.7.1.

Because the GAP model and its solver are a black-box to our strategies, their optimization is not in the scope of our work. To ensure the solution quality with a reasonable time-contract for the solver, we compared it on a set of sample instances with mulknap,[1] an exact MKP solver (Pisinger 1999). With a 60 second timeout, CPLEX achieves on average 99.5 % of the optimal solution calculated by mulknap.

All strategies are run on each scenario with a 60 second timeout for the GAP solver. The thresholds γ for the Objective Switch strategy are 10, 20, 30, and 40. We evaluate the full rotation of tasks over agents, both looking at all tasks, and at each individual task. One full rotation over all tasks is achieved, when each task was assigned once to all compatible agents. The rotation over one task reflects how often a task in average is assigned to its compatible agents. These numbers can be different. If few tasks are not rotated, those forestall full rotations, but still allow other tasks to be frequently rotated.

Furthermore, we compare the achieved profit of the assignments with the profit of the FOP strategy, which does not consider rotation and only maximizes profit. As the other experimental parameters are the same and also the same assignment model is used, FOP simulates the baseline setting without rotation-awareness.

We have considered an additional baseline, where the full multi-cycle assignment problem is optimized as one single optimization model. This differs from our method, as all availabilities are known upfront. However, due to the exceeding model size, solving the extended GAP model is computationally expensive and did not yield a comparable solution within 24 CPU hours, which is substantially more than the total computational cost of successively optimizing individual cycles. Therefore, we do not further consider this baseline.

5.2 Multi-Cycle Multiple Knapsack Problem

MKP is a variant of the 0-1 knapsack problem, and thereby of GAP, with multiple agents (knapsacks) (Pisinger 1995).

We extend MKP to a multi-cycle variant (MCMKP) with limited availability of tasks and agents. In every cycle, the same MKP instance has to be solved under consideration of the assignments made in previous cycles and changing availability of tasks and agents.

To generate problem instances, we follow the procedure by Pisinger (1999), as described in Fukunaga (2011), and extend

---

[1]http://www.diku.dk/~pisinger/codes.html

		Avail. $\mathcal{A}$	75%	75%	100%	100%
	Avail. $\mathcal{T}$	75%	100%	75%	100%
Strategy	FOA	2.3 (4.5)	2.0 (4.8)	3.0 (5.8)	3.3 (6.2)
	OS/10	**2.0 (4.4)**	**2.0 (4.7)**	**3.0 (5.3)**	**3.3 (5.9)**
	OS/20	1.7 (3.9)	2.0 (4.4)	1.7 (4.0)	3.0 (5.2)
	OS/30	1.0 (3.2)	1.7 (3.9)	1.3 (3.4)	2.3 (4.4)
	OS/40	1.0 (2.8)	1.3 (3.4)	0.7 (3.0)	1.7 (3.8)
	PC	0.0 (4.3)	0.0 (4.5)	0.0 (5.5)	0.0 (5.9)
	WPP	1.3 (4.0)	1.3 (4.3)	1.3 (4.9)	1.7 (5.3)
	FOP	0.0 (1.7)	0.0 (1.5)	0.0 (1.9)	0.0 (2.0)

Table 1: Rotational Diversity for MCMKP: Full rotations of all tasks (Avg. rotations per task; Bold: Best, without FOA/FOP baselines).
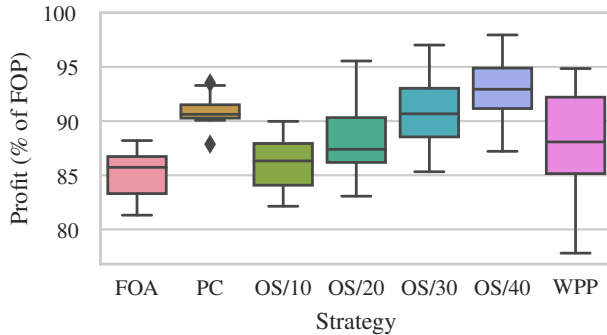

Figure 4: MCMKP: Profits in comparison to baseline (% of FOP). The effect of different thresholds for OS/γ are clearly visible.

it to the notion of compatibility and availability. Task weights are drawn from a uniform distribution ($w_j \sim \mathcal{U}[10, 1000]$), and the profits are either uncorrelated or weakly correlated. In the former case, profits are drawn from the same uniform distribution. In the latter case, the profits are calculated by $p_j = w_j + \mathcal{U}[-99, +99], \forall j \in \mathcal{T}^k$. The agents' capacities are set to 40–60 % of the tasks' weight, except for the last agent, whose capacity is set such that the total capacity equals half of the tasks' demand. The instance sizes are 30/75, 15/45, and 12/48 agents, respectively tasks. For this generation scheme, a ratio $|\mathcal{T}^k|/|\mathcal{A}^k|$ slightly larger than 2 leads to hard instances, while instances of higher ratios become easier to solve (Fukunaga 2011). The number of cycles is three times the number of tasks, to allow multiple assignments between tasks and agents, even if an agent has only capacity for one task.

A notion of compatibility is implicit in the generation procedure. Tasks that do not fit into an agent's capacity are automatically incompatible. However, this skews the number of compatible tasks to those agents with high capacities, and puts more emphasis onto their assignments.

We take all combinations of the four parameters to generate 24 instances. The results are shown in Table 1 and Figure 4. We look at the rotational diversity results grouped by agent and task availability, that is in four different groups, as this is the main differentiating attribute of the MCMKP scenarios.

The rotation-aware strategies, that balance profit maximization and rotation, rank in-between the baseline strategies FOA and FOP, both regarding profit and rotation.

More specifically, the OS/γ strategies, which switch maximizing profit and diverse assignments based on an AP threshold, are particularly effective in the MCMKP setting. With its low γ value, OS/10 performs similar to FOA, as it more frequently focuses on diverse assignments, but still achieves slightly better profits, whereas OS/40, with its high γ, has the highest profit of the strategies, with ¿90 % profit compared to FOP, but better rotation than the FOP baseline.

Product Combination, the multiplication of profits and affinities, is not particularly effective in the MCMKP setting. Although it leads to a high profit utilization and a higher rotation per task than the other strategies, it is not capable to achieve full rotation. This shows, that there are a few tasks that do not receive high enough task values by the PC strategy, such that they are lucrative enough to be assigned in the GAP. As noted before, the MCMKP generation procedure shifts the compatibility of tasks to the last agent, to which most tasks are compatible. Still, its capacity is limited and only a few tasks, especially those with a high weight, can be assigned per cycle. This bottleneck in the diverse assignment process can hinder full rotations. Nevertheless, as other strategies are capable to overcome the bottleneck and can achieve full rotations, this is a negative aspect of the PC strategy.

The WPP strategy maintains a good level of average rotations per task and a medium level of full rotations. However, the profit trade-offs are not consistent, but vary between the different scenarios, which makes WPP unsuitable as a general strategy, compared to PC and OS/γ.

Furthermore, we analyze the influence of varying availability on the achievable rotations. As the setting is such that a selection of tasks has to occur (the resource demand is higher than the resource supply), the availability of a large number of agents has a stronger influence than a high task availability.

However, for making diverse assignments, a high task availability is beneficial. This can be seen when comparing the results with 75 %/100 % and 100 %/75 % agent respectively task availability. The more profit-oriented variants of OS/γ achieve better rotation in the former than in the latter case, as OS/γ switches focus, and potentially the optimization objective of a cycle does not match the availability of the tasks. Then, one task might only be present at profit maximization, but not for rotation optimization.

5.3 Test Case Selection and Assignment

As a second case study, we employ the real-world application of Test Case Selection and Assignment (TCSA) for cyber-physical systems (Yoo and Harman 2012), such as industrial robots. TCSA usually occurs in Continuous Integration (CI) processes, where new releases of the robot control software are regularly integrated and released (Mossige et al. 2017). Typically, CI involves assigning test cases to test agents several times a day. Comprehensive test suites exist, but available time and hardware for their execution are limited. Then it is necessary to distribute a selection of the most relevant test cases over the available agents. The test case relevance is given by an upstream test case prioritization process. This

Strategy	20 / 750	20 / 1500	20 / 3000	30 / 3000	Total
FOA	96.1	79.0	67.4	74.4	79.2
OS/10	96.3	79.5	67.8	75.0	79.7
OS/20	98.2	80.0	68.3	75.4	80.5
OS/30	99.1	85.0	69.0	75.9	82.3
OS/40	99.6	90.9	69.6	76.6	84.2
PC	**99.7**	**97.7**	**91.1**	**96.6**	**96.3**
WPP	98.1	77.4	54.6	66.6	74.2
FOP			100.0		

(a) Profit in comparison to baseline (% of FOP)

Strategy	20 / 750	20 / 1500	20 / 3000	30 / 3000	Total
FOA	15 (24.4)	6 (15.7)	3 (9.5)	3 (8.5)	27 (14.5)
OS/10	14 (22.2)	6 (15.5)	**3 (9.4)**	**3 (8.4)**	26 (13.9)
OS/20	9 (18.6)	6 (15.3)	3 (9.2)	3 (8.3)	21 (12.9)
OS/30	7 (16.9)	5 (14.3)	3 (9.1)	3 (8.1)	18 (12.1)
OS/40	7 (16.2)	4 (13.1)	3 (8.9)	3 (7.9)	17 (11.5)
PC	**15 (24.0)**	**7 (14.4)**	3 (8.3)	3 (7.5)	**28 (13.6)**
WPP	14 (24.1)	7 (14.2)	3 (7.3)	3 (7.0)	27 (13.2)
FOP	3 (15.7)	0 (10.8)	0 (7.1)	0 (4.6)	3 (9.6)

(b) Diversity: Full rotations of all tasks (Avg. rotations per task)

Table 2: Rotational Diversity for TCSA: The PC strategy is most effective. (Bold: Best, not considering FOA/FOP baselines)

priority can be different at each cycle, due to discovered failures or changes in the system-under-test. The assignment of tests to agents is constrained by the available time and compatibility between test and agent. In the GAP terminology, the test case priority resembles the profit, the test's runtime the task weight, and an agent's available time its capacity.

Additionally, the availability of agents is influenced by maintenance, technical faults, or short-term usage in other projects, and the set of test cases changes due to the ongoing development. Therefore, TCSA cannot be solved by a static assignment without the need for frequent updates. Instead, to capture the dynamic setting, an individual selection and assignment has to be made at each cycle. Enforcing diverse assignments increases the coverage of tasks and agents, and thereby the confidence into the system-under-test.

We evaluate the strategies on TCSA, based on actual test data from our industrial partner. All test agents have the same capacity, that is the time available for a test cycle (10 hours). Due to unique hardware specifications and different functionalities, a test case is compatible with approximately 60 % of the test agents. The runtime of a test case varies from 1 to 21 minutes, but is identical for each test agent. In practice, test agents are not exclusively available for testing and test cases are temporarily removed from the test suite. Therefore, an average of 40 % of the agents and 10 % of the test cases are unavailable for 3–7 cycles.

We consider four scenarios, 20 agents with 750, 1500, and 3000 test cases, and 30 agents with 3000 test cases. Table 2 summarizes the results. All strategies, except FOP, which is rotation-unaware, are able to maintain rotational diversity at a similar level, regarding both the full rotations of tasks, and the average number of rotations per task.

In the smallest scenario, a full rotation of all tests over all possible agents is achieved 14–16 times over 365 cycles, i.e. every 23–26 days. Here, each task is compatible to ˜12 agents (60 %), and 60 % of the agents are unavailable for multiple cycles. For the larger scenarios with the same number of agents, the number of full rotations reduces approximately linear, but not the number of average rotations per task. This shows, that some tests are not evenly rotated and hinder the completion of full rotations. With a larger number of agents, the average number of rotations per tasks drops, as there are more compatible agents and it takes more cycles to rotate.

The profits earned from the assignments (see also Table 2) are close to the FOP baseline in the smallest scenario, but decrease with a higher number of tests, except for PC, which is able to balance profit maximization and rotation better than the other strategies and even outperforms FOA for complete rotations. For PC, the profit trade-off is always $< 10\,\%$, and on average $< 4\,\%$ in comparison to the profit-oriented FOP.

6 Conclusion

Rotational diversity is the frequent assignment of a task to all its compatible agents over subsequent cycles. We present a two-part model for its optimization in multi-cycle assignment problems with variable availability of tasks and agents: 1) an inner assignment problem, to optimize the assignment from tasks to agents, and 2) an outer problem, to adjust the task values for the maximization objective of the inner problem.

Five strategies, each having a different approach and trade-offs, are evaluated on two case studies. Achieving rotational diversity is possible with a profit trade-off of only 4 % in the test case selection and assignment case study. Both the product combination of profits and affinities, and the objective switch strategy, that focuses on either profit maximization or diverse assignments, efficiently achieve rotational diversity.

For applications of this method, we encourage the reader to start from the product combination strategy. It is simple to implement and does not require initial configuration, but allows to be adjusted if necessary.

The combination of profits and affinities into a single task value is efficient for balancing profits and rotation. This is especially the case in settings where an extended multi-objective optimization model is not an alternative. Splitting the problem and its responsibilities allows to use problem-specific, single-objective solvers for the inner problem, or to use problems with additional requirements, e.g. precedence constraints or task-dependencies.

References

Anshelevich, E.; Chhabra, M.; Das, S.; and Gerrior, M. 2013. On the Social Welfare of Mechanisms for Repeated Batch Matching. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 60–66.

Ayough, A.; Zandieh, M.; and Farsijani, H. 2012. GA and ICA approaches to job rotation scheduling problem: Considering employee's boredom. *International Journal of Advanced Manufacturing Technology* 60(5-8):651–666.

Azizi, N.; Zolfaghari, S.; and Liang, M. 2010. Modeling job rotation in manufacturing systems: The study of employee's boredom and skill variations. *International Journal of Production Economics* 123(1):69–85.

Bard, J. F., and Purnomo, H. W. 2005. Preference scheduling for nurses using column generation. *European Journal of Operational Research* 164(2):510–534.

Bertsimas, D.; Natarajan, K.; and Teo, C.-P. 2006. Persistence in discrete optimization under data uncertainty. *Mathematical Programming* 108(2-3):251–274.

Bhadury, J., and Radovilsky, Z. 2006. Job rotation using the multi-period assignment model. *International Journal of Production Research* 44(20):4431–4444.

Brown, G. G.; Dell, R. F.; and Wood, R. K. 1997. Optimization and Persistence. *Interfaces* 27(5):15–37.

Chiaramonte, M. V. 2008. Competitive nurse rostering and rerostering. *ProQuest Dissertations and Theses* 3295196(May):154.

Clarke, L.; Johnson, E.; Nemhauser, G.; and Zhu, Z. 1996. The aircraft rotation problem. *Transportation Research Part A: Policy and Practice* 30(1):51.

de Vries, S., and Vohra, R. V. 2003. Combinatorial Auctions: A Survey. *INFORMS Journal on Computing* 15(3):284–309.

Endriss, U.; Maudet, N.; Sadri, F.; and Toni, F. 2006. Negotiating socially optimal allocations of resources. *Journal of Artificial Intelligence Research* 25:315–348.

Ernst, A. T.; Jiang, H.; Krishnamoorthy, M.; and Sier, D. 2004. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research* 153(1):3–27.

Faaland, B. H. 1981. Technical Note–The Multiperiod Knapsack Problem. *Operations Research* 29(3):612–616.

Fukunaga, A. S. 2011. A branch-and-bound algorithm for hard multiple knapsack problems. *Annals of Operations Research* 184(1):97–119.

Glover, F.; Løkketangen, A.; and Woodruff, D. 2000. Scatter search to generate diverse MIP solutions. *OR Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research* 299–317.

Hebrard, E.; Hnich, B.; O'Sullivan, B.; and Walsh, T. 2005. Finding diverse and similar solutions in constraint programming. In *Proceedings of the Twentieth AAAI National Conference on Artificial Intelligence*, 372–377.

Hosseini, H.; Larson, K.; and Cohen, R. 2015. Matching with Dynamic Ordinal Preferences. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 936–943.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer. 85–103.

Liu, Y., and Mohamed, Y. 2008. Multi-agent resource allocation (MARA) for modeling construction processes. In *Simulation Conference, 2008. WSC 2008. Winter*, 2361–2369.

Ma, Y.; Chu, C.; and Zuo, C. 2010. A survey of scheduling with deterministic machine availability constraints. *Computers and Industrial Engineering* 58(2):199–211.

Martello, S., and Toth, P. 1987. Algorithms for Knapsack Problems. *North-Holland Mathematics Studies* 132(C):213–257.

Morrison, T. 2010. a New Paradigm for Robust Combinatorial Optimization: Using Persistence As a Theory of Evidence. *PhD thesis*.

Mossige, M.; Gotlieb, A.; Spieker, H.; Meling, H.; and Carlsson, M. 2017. Time-Aware Test Case Execution Scheduling for Cyber-Physical Systems. In *Proceedings of the 23rd International Conference on Principles and Practice of Constraint Programming*, volume 10416 of *LNCS*, 387–404. Springer.

Musliu, N.; Schutt, A.; and Stuckey, P. J. 2018. Solver Independent Rotating Workforce Scheduling. In *CPAIOR*, 429–445.

Nethercote, N.; Stuckey, P.; Becket, R.; Brand, S.; Duck, G.; and Tack, G. 2007. MiniZinc: Towards a standard CP modelling language. In Bessiere, C., ed., *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*, volume 4741 of *LNCS*, 529–543.

Nongaillard, A.; Jaumard, B.; Mathieu, P.; and Groupe D'études Et De Recherche En Analyse Des Décisions (Montréal, Q. 2009. A multi-agent resource negotiation for the utilitarian social welfare. *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology* 1–16.

Pentico, D. W. 2007. Assignment problems: A golden anniversary survey. *European Journal of Operational Research* 176(2):774–793.

Petit, T., and Trapp, A. C. 2015. Finding Diverse Solutions of High Quality to Constraint Optimization Problems. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, 260–266.

Pisinger, D. 1995. *Algorithms for Knapsack Problems*. Ph.D. Dissertation.

Pisinger, D. 1999. An exact algorithm for large multiple knapsack problems. *European Journal of Operational Research* 114(3):528–541.

Sandholm, T., and Suri, S. 2003. BOB: Improved winner determination in combinatorial auctions and generalizations. *Artificial Intelligence* 145(1-2):33–58.

Trapp, A. C., and Konrad, R. A. 2015. Finding diverse optima and near-optima to binary integer programs. *IIE Transactions* 47(11):1300–1312.

Yoo, S., and Harman, M. 2012. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability* 22(2):67–120.