Weighted Capacitated Popular Matching for Task Assignment in Multi-Camera Networks

Lin Cui*† and Weijia Jia†
*Department of Computer Science, Jinan University, Guangzhou, China
†Department of Computer Science, City University of Hong Kong, HK SAR, China
Email: {lincui2,wei.jia}@cityu.edu.hk

***Abstract*—Multi-Camera Networks (MCN) are becoming increasingly important in today's society needs and daily-life with application-oriented multiple tasks running in each camera such as video surveillance, object tracking and localization etc. The ultimate goal of MCN is to best satisfy such tasks' preferences/expectations required by users, which has not been well-addressed by previous works. This paper investigates such challenge by formulating a novel weighted capacitated Popular Matching for multi-Task assignments (PMT) problem and proposing efficient algorithms to solve the problem. Using the popularity to represent the optimality of task-camera matching, we can find a matching in which the allocation of the most tasks to the corresponding cameras is closest to the tasks' preferences. With extensive simulations, we demonstrate that our approaches can make matching to the satisfaction of all tasks efficiently as compared to those baseline approaches.**

I. Introduction

A Multi-Camera Network (MCN) is a combination of embedded systems, wireless sensor networks (WSN) and video transmission and processing. It is expected to gain a wide proliferation in the near future in the sense that huge number of (potential high-definition) wired/wireless cameras will be installed worldwide by enterprises and home users for smart business and smart home applications.

A camera network allows many compelling applications such as object tracking, event detection or video surveillance [1]. In most practical camera systems, resources are usually constrained, which may arise from network bandwidth, I/O, memory and CPU capability [2]. Normally, a camera needs to perform many tasks, where we refer to a task as the operation request by users or applications for at least one camera. Due to the restricted resources, the capacity of how many tasks can be executed simultaneously for a camera is limited. According to users priority and task nature, we consider that different tasks are prioritized, e.g., the task of *routine patrolling* may have the lowest priority, while *tracking* tasks that are triggered by motion detection scheme may need to be performed immediately. In addition to the priority, tasks in MCN are also time-sensitive. Furthermore, these tasks have different preferences on the set of acceptable cameras. The preference lists can be determined by many ways, e.g., location of cameras or cross correlation of motion amongst cameras for some tracking applications [2]. A well coordination among different tasks and cameras, considering all these issues above, is crucial to the success of a MCN system.

Among all the acceptable cameras, a task should always be assigned the best possible one to maximize the benefit of the system. At the same time, there is a deadline for each task, after which the task would be obsoleted. Such real-time restrictions are particularly limited because of the enormous computational demands and the resource limitations imposed by MCN. Scheduling under resource constraints has been previously studied in a variety of contexts [3], [4]. While those previous works are related to the allocation problem of cameras, none of them applies directly, due to the unique aspects of real-time camera network setting. Cenedese et al. [5] studied a similar problem of task assignment in multi-agent multi-task systems, especially for smart camera networks for surveillance. However, they only focus on the simple one-to-one case where each camera can only perform one task, which is not realistic for most practical MCN systems.

In this paper, we consider that popularity is important for the preference matching. Popular matching [6] is a strong definition of optimality trying to find a matching in which most of tasks are assigned to cameras that are closest to their preference. In terms of weighted tasks assignment, a popular matching is normally defined as the assignment, in which, the total weight of tasks which prefer such assignment is larger than that of any other allocations. A popular allocation should be preferred by most tasks, satisfying their preference as much as possible while considering their weight. Therefore, we focus on the resource constrained scheduling in the setting for MCN, studying the popularity for multi-task assignment, which, to the best of our knowledge, has not been studied for MCN previously. Our goal is to design a principled extensible strategy for the task assignment problem in MCN.

II. Multi-Task Assignments for MCN

In MCN, there are m active cameras $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ distributed in the environment and connected to the network through either Ethernet or WLAN. All cameras are attached to a server which is responsible for the maintenance of the network and schedule of tasks from users. For each camera c_i, its hardware capability and network bandwidth resource restrict the maximum number of tasks to be executed at the same time, i.e., the capacity of the camera, denoted by s_i.

A set of n tasks that can be issued to the network is $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$. We assume all these tasks are asynchronous, i.e., they do not communicate with each other. And for simplicity, suppose each task only requests one camera. The set $\mathcal{T}$ will be changed with time, as new tasks can arrive and existing tasks can complete or be dropped. Without loss of generality, we characterize a task from the following aspects: (a) *Type*: A priority is assigned to tasks of each type,

i.e., $Pr(t_i) \in (0, 1]$ for task t_i; (b) *Target*: The target of a task can be an area to be monitored; (c) *Service Time*: The total time $T_{service}$ required to complete the execution of the task; (d) *Drop Time*: The time T_{drop} after which the task will be considered obsolete; (e) *Weight*: Considering both the type of the task and the drop time, the weight of task t_i is $w(t_i) = f(Pr(t_i), T_{drop}) = \alpha \times Pr(t_i) + (1 - \alpha) \times (T_{curr} - T_{occ})/(T_{drop} - T_{occ})$, where T_{curr} is the current time and T_{occ} is time when t_i occur.

The monitored environment of the MCN is partitioned into p areas of interest, which are coincide with the field of views (FOV) of at least one camera. Each camera covers one or more areas. So it is possible to build a coverage matrix $V(i, j)$ by defining element v_{ij} to represent the distance of camera c_i to the monitor area p_j.

In particular, according to the services to be provided by MCN, we design the following set of task types: *Routine Patrolling (RP)*, *Streaming Service (SS)*, *Automatic Tracking (AT)* and *User-indicated Task (UT)*. The *RP* tasks are the basic service to be provided by all cameras. They are assigned with the lowest priority, and infinite service time, because *RP* tasks represent the default state of the surveillance system. For task $t_i \in \mathcal{T}$, a list $L(t_i)$ of all fulfilled cameras in $\mathcal{C}$ can be built. All cameras in $L(t_i)$ are ranked in descending order. We suppose the set $\mathcal{T}$ is fixed for a specific time point.

A matching $M = \{(t_i, c_j) | c_j \in L(t_i), t_i \in \mathcal{T}, c_j \in \mathcal{C}\}$ is defined as a subset of pairs from $\mathcal{T} \times \mathcal{C}$. Denote $M(t_i)$ to be the camera that t_i is assigned to execute, and $M(c_j)$ to be the set of tasks that are assigned to c_j. For a list of $\mathcal{T}$ and $\mathcal{C}$, following are some constraints need to be considered: 1) *Acceptable Pairs*: The first constraint arise by the coverage matrix V is that each pair in the assignment must be acceptable. Only cameras from $L(t_i)$ can be assigned to execute t_i. A pair of task and camera (t_i, c_j) is called *acceptable* if $c_j \in L(t_i)$. 2) *Capacity Constraints*: Each task should be assigned to at most one camera, i.e., $|M(t_i)| \leq 1$. For each camera, the capacity size limits the maximum tasks that can be executed simultaneously, i.e., $|M(c_j)| \leq s_j$. 3) *Time Constraints*: Tasks in MCN are usually time-sensitive. A task will be considered obsolete and be dropped in T_{drop} after it is issued to MCN.

In addition to the constraints above, we also focus on the *popularity* of a matching: Let $P(M_1, M_2)$ denote the set of tasks who prefer M_1 to M_2 according to their preference lists. The *satisfaction* of M_1 with respect to M_2 is defined as:

$$sat(M_1, M_2) = \sum_{t_i \in P(M_1, M_2)} w(t_i) - \sum_{t_i \in P(M_2, M_1)} w(t_i) \quad (1)$$

A higher *satisfaction* of M_1 over M_2 means that the assignment of M_1 is more optimal than M_2 from the viewpoint of tasks. We say that M_1 is *more popular than* M_2 if $sat(M_1, M_2) > 0$. Then the *popular matching* is defined as:

Definition 1. *A matching M is called a* popular matching *if there is no other matching that is more popular than M.*

In the scenario of MCN with multi-tasks, we intend to find an optimal allocation that satisfies all the constraints above, in accordance with the preference and weights of tasks.

Definition 2. *The weighted capacitated Popular Matching for multi-Task Assignment (PMT) problem in MCN: For an Instance I, the PMT problem is to find a popular matching M with maximum cardinality:*

$$\begin{aligned} &\max |M| \\ &s.t.\ |M(c_i)| \leq s_i \\ &\quad |M(t_j)| \leq 1 \end{aligned} \quad (2)$$

where $i = 1, 2, \ldots, n$, and $j = 1, 2, \ldots, m$.

Task (Weight)	Preference List	Camera (Capacity)
t_1 (0.25):	c_2, c_3	c_1 (2)
t_2 (0.5):	c_4, c_1, c_3	c_2 (1)
t_3 (0.75):	c_3, c_2	c_3 (2)
t_4 (0.6):	c_1, c_4	c_4 (1)
t_5 (0.4):	c_1, c_4	
t_6 (0.3):	c_3, c_2	

Fig. 1: An instance of PMT problem

For exposition purposes, we create a unique virtual *last resort* camera $l(t_i)$ for each $t_i \in \mathcal{T}$ and append $l(t_i)$ to t_i's preference list $L(t_i)$ as the last element. The *last resort* $l(t_i)$ is unique for every task, which assures that all tasks can be matched. If t_i is matched to $l(t_i)$ in M, it means t_i is unassigned to any cameras. So the PMT problem is equal to finding a task-complete popular matching reducing the number of matched $l(t_i)$ for all $t_i \in \mathcal{T}$. Figure 1 gives an example of PMT problem.

III. Algorithms for Matching Tasks and Cameras

A. Overview

Consider PMT basic operations in a MCN system, for convenient of tasks management, two lists are defined: *Waiting Task List* (*WTL*) and *Running Task List* (*RTL*). Therefore, the whole task pool $\mathcal{T}$ is equal to $WTL \cup RTL$. As we have explained before, the length of the task pool is dynamic as new tasks can occur and old ones can complete or be dropped. When a new task t_i arrives, it is initially added to *WTL*, and then it is either moved to *RTL* if a camera is assigned to execute t_i or it is dropped off if it becomes obsolete after $T_{drop}(t_i)$. Specially, suppose a *RP* task is the default task for each camera, it will never be dropped and will only transit between *WTL* and *RTL* after it is issued for a camera.

To avoiding situations in which a task is temporarily paused and moved from *RTL* to *WTL*, or it is frequently switching among different cameras before the end of the execution, we first set a threshold T_{th}. When updating the task set $\mathcal{T}$ for a new matching, a task is not allowed to be moved out of *RTL* when $T_{drop} - T_{curr} \leq T_{th}$, where T_{curr} is the current time. In the meantime, when a new task t_i is issued to the network, only tasks with equal or lower priority in *RTL* are used for re-computation of the popular matching, i.e., $Pr(t_j) < Pr(t_i), \forall t_j \in RTL$.

B. Popular PMT Matching

To solve the PMT problem, we revise to use the WCHA (Weighted Capacitated House Allocation) algorithm [7] to find a maximum popular matching, where tasks play the role of agents and cameras as houses in WCHA problem.

$\mathcal{T}$ is partitioned into sets $W_1, W_2, \ldots, W_{N_w}$, such that tasks in W_i have the same weight w_i and $w_1 > w_2 > \ldots > w_{N_w} > 0$, where N_w is the number of these sets. For each task t_i, introduce the notion of t_i's *f-camera* and *s-camera* denoted by $f(t)$ and $s(t)$ respectively. Let $f_{i,j}$ be the number of tasks in W_i whose f-camera is defined and equal to c_j. Given $1 \leq z \leq N_w$, for every task $t_i \in W_z$, let $f(t)$ be the most preferred camera c_j on $L(t_i)$ such that $\sum_{x=1}^{z-1} f_{x,j} < s_j$. Specially, for $t_i \in W_1$, $f(t_i)$ is the first-ranked camera on $L(t_i)$.

For each $c_j \in C$, let $f(c_j) = \{t | f(t) = c_j, t \in \mathcal{T}\}$ and $f_j = |f(c_j)|$, i.e., $f(c_j) = \bigcup_{x=1}^{N_w} f_p(c_j)$. We call c_j an f-camera if $f_j > 0$. For each f-camera, a maximum weight level d_j is defined as follows:

$$d_j = \begin{cases} max\{z : 0 \leq z \leq N_w \wedge f_{z,j} > 0\}, & f_j \leq s_j \\ max\{z : 0 \leq z \leq N_w \wedge \sum_{x=1}^{z} f_{x,j} < s_j\}, & f_j > s_j \end{cases} \quad (3)$$

d_j represents the maximum weight level of tasks that c_j can accept. Given $1 \leq z \leq N_w$, for every task $t_i \in W_z$, let $s(t)$ be the most preferred camera c_j on $L(t_i)$ such that $c_j \neq f(t_i)$ and $\sum_{x=1}^{z} f_{x,j} < s_j$. It is possible that $s(t_i) = \emptyset$ if $f(t_i) = l(t_i)$.

According to WCHA [7], we can get the necessary conditions for a matching to be popular in an instance of PMT, i.e., every task t is matched to either $f(t)$ or $s(t)$ in a popular matching. The *PMT-Popular-MAX* algorithm to find a maximum weighted popular matching is given in Algorithm 1.

In the *PMT-Popular-MAX* algorithm, we consider a graph G' containing all $\mathcal{T} \cup \mathcal{C}$ and two edges for each $t \in \mathcal{T}$, i.e., $(t, f(t))$ and $(t, s(t))$. It follows that all popular matchings must be contained in G' according to the necessary conditions for a popular matching. Next, we will apply the *Prune-WCHA()* [7] function to remove certain edges in G' that can not be part of any popular matching. In line 13, we refer to the Gabow's algorithm [8] to compute the maximum matching M' in G''. The *PMT-Popular-MAX* algorithm can find a maximum popular matching or determine that none exists in $O(\sqrt{S}n+l)$ time, where $S = \sum_{j=1}^{m} s_j$ is the total capacities of all cameras and $l = \sum_{i=1}^{n} |L(t_i)| \leq nm$ is the total preference list length of all tasks. The detailed operations and proofs of correctness of the *PMT-Popular-MAX* are similar to [7].

C. Analysis

A popular matching need not always exist, though Abraham et al. [6] presented empirical evidence that these cases are rare when preference lists are independent. In case that the *PMT-Popular-MAX* algorithm does not admit any popular matching, we intend to compute a *weighted rank-maximal matching* [9] for the PMT problem, which matches the maximum weight of tasks to their first preferred cameras in $L(\cdot)$, and subject to this, the maximum weight to their second preferred cameras, and so on. The time to compute such a matching is $O(min(\gamma\sqrt{n+m}, n+m)l)$, where $\gamma = N_r \times N_w$, N_r is the maximum rank and $l = |\mathcal{E}|$ is the total number of all edges, i.e., the length of all tasks' preference lists [9], [10].

Considering the overall complexity, if the *PMT-Popular-MAX* finds a popular matching, it will be returned in $O(\sqrt{S}n+l)$ time. Otherwise, *PMT-Popular-MAX* will be at least terminated at line 14 to indicate that no popular matching exists.

Algorithm 1 PMT-Popular-MAX

Input: $\mathcal{T}$, $\mathcal{C}$
Output: A popular matching M

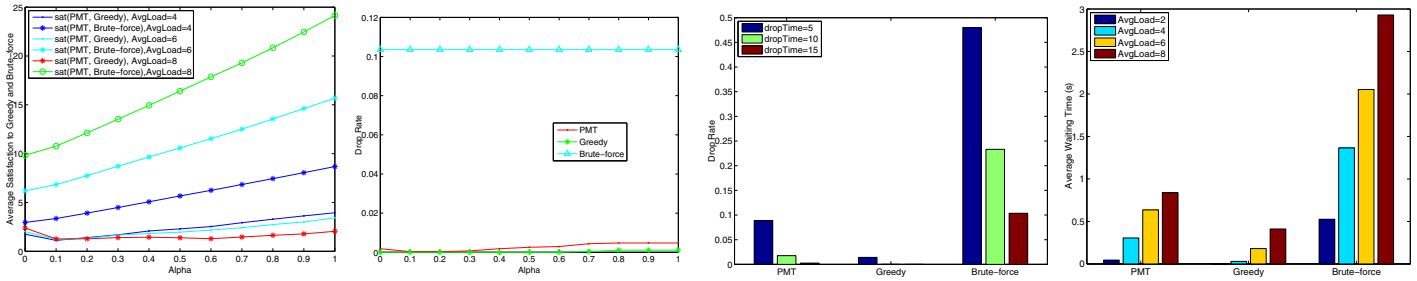
1: $M = \emptyset$
2: Compute $f(t_i)$ and $s(t_i)$ for each task $t_i \in T$
3: $G' = (\mathcal{T} \cup \mathcal{C}, E = \{(t_i, f(t_i)), (t_i, s(t_i)) | \forall t_i \in T\})$
4: G''=Prune-WCHA(G')
5: $A_1 = \{t | s(t) = l(t), t \in \mathcal{T}\}$
6: $A_2 = \mathcal{T} \setminus A_1$
7: $A' = \{t | t \in \bigcup_{x=1}^{d_j} f_x(c_j), c_j \in \mathcal{C}\}$
8: $M_0 = \emptyset$
9: **for each** $t_i \in A'$ **do**
10: $\quad M_0 = M_0 \cup \{(t_i, f(t_i))\}$
11: **end for**
12: Remove all isolated and full cameras, including incident edges, from G''
13: Compute a maximum matching M' in G'' involves only $A_2 \setminus A'$
14: **if** M' is task-complete in G'' **then**
15: $\quad$ Remove all edges incident to last resort
16: $\quad$ Augment M' for $A \setminus A'$ tasks
17: $\quad M = M_0 \cup M'$
18: $\quad$ Assign all $t \in A \setminus A'$ that $M(t) = \emptyset$ to $l(t)$
19: $\quad$ **repeat**
20: $\quad\quad$ **if** some $t_i \in A \setminus A'$ that $f(t_i) \neq M(t_i)$ and $|M(f(t_i))| < s_j$ **then**
21: $\quad\quad\quad$ Promote t_i from $M(t_i)$ to $f(t_i)$ in M;
22: $\quad\quad$ **end if**
23: $\quad$ **until** nothing changes
24: **end if**
25: Output matching M

The *Prune-WCHA()* cost $O(l)$ time, computing a maximum matching on the reduced graph G'' in line 13 cost at most $O((n+m)\sqrt{l})$ time. Then, in case that a popular matching does not exist, PMT needs $O((m+n)l)$ time to output a weighted rank-maximal matching.

IV. Evaluations

A. Simulation Settings

We consider a scenario which is divided into many subareas to be monitored. Each area is covered by at least two cameras. The capacity of each camera is varied between 2∼4. The coverage matrix V, which is used to compute each task's preference list, is generated randomly. We consider four different basic task types: *RP*, *SS*, *AT* and *UT*, which are explained in Section II. The value of priority assigned each task type are: $Pr(RP) = 0.25, Pr(SS) = 0.5, Pr(AT) = 0.75, Pr(UT) = 1$, i.e., *RP* tasks have the lowest priority while *UT* task have the highest priority. In the initial state of each simulation, we assign each area a routine patrolling task *RP*, which is very normal for video surveillance applications in MCN. Those *RP* tasks have infinite service time. The factor of Average Load (*AvgLoad*) is used to represent how many tasks in {*SS*, *AT*, *UT*} occurs in each second of simulation. The targets of those tasks are randomly distributed among all areas. The service time and drop time of each task are random generated. For convenience, we consider *dropTime* = $T_{drop} - T_{occur} - T_{service}$ to be varied among {5s, 10s, 15s}.



(a) Average Satisfaction on α and average load: dropTime = 10

(b) Drop Rate on α: AvgLoad=6, *dropTime*=10

(c) Drop Rate on *dropTime*: AvgLoad=6, $\alpha = 0.5$

(d) Average waiting time on average load: $\alpha = 0.5$, *dropTime*=10

Fig. 2: Performance comparision

The following two baseline approaches from [5] are introduced and revised to compare with PMT: 1) *Greedy Assignment*: In the greedy approach, each camera $c_j \in \mathcal{C}$ will periodically scan the *WTL* for the best acceptable task t' in terms of weight. If t' has higher weight than those which are currently executed by c_j, c_j will start executing t', dropping the worst tasks to the *WTL*. Otherwise, c_j keeps executing its current tasks. 2) *Nearly Brute-force*: Its underlying idea is that: when a new task t_i occurs (other than *RP* task), the first acceptable camera that can accept t_i (either free or it is executing a lower weighted task) is assigned to execute t_i. If multiple cameras are free, just choose the most preferred one in $L(t_i)$. This approach assures maximum continuity in matching tasks and cameras, however, it does not give any guarantees on the quality of assignment.

B. Results and Analysis

In addition to satisfaction defined in Equation (1), we also study the metrics of *Average Waiting Time*, which is the average time of all completed tasks spending in *WTL*, and *Dropping Rate*, which is the ratio of all dropped tasks to the number of all issued tasks.

Figure 2 shows the evaluation results. Figure 2a is the PMT satisfaction with respect to both Greedy and Brute-force approaches on α under different *AvgLoad*, where the *dropTime* is 10s. The PMT approache outperforms the other methods. Figure 2b and 2c show the drop rate of all the three methods. The Brute-force approach always has the highest drop rate. For PMT and Greeedy approaches, drop rate is slight increased with higher α. This is because when α is increased, the contribution of weight from the lifetime of tasks becomes smaller.The performance of average waiting time is shown in Figure 2d. For all the three methods, the average waiting time is increasing with the average load due to more intense competition among tasks. Specially, the Greedy approach has the smallest waiting time, while the Brute-force method suffers the longest delay.

V. Conclusions

In this paper, we study the multi-task assignment problem in Multi-Camera Networks (MCN) and introduce the popular matching to solve such schedule issue. The optimality of the allocation is represent by the popularity of a matching, which maximize the weight of tasks assigning to their closest preference. We model the weighted capacitated Popular Matching for multi-Task assignment (PMT) prblem in MCN, considering both the weight of different types of tasks and their preferences. An efficient algorithm is proposed, which can either output a popular matching in $O(\sqrt{S}n+l)$ time or a weighted rank-maximal matching in $O((m+n)l)$ time in case that a popular matching does not exist.

Acknowledgment

This work is supported in part by grants from the General Research Fund of the Hong Kong SAR, China No. (CityU 114012, CityU 114513); and CityU Applied R & D Grants (ARD) No. 9681001; (ARG) No. 9667052; NSF (China) No. 61070222 and Shenzhen (China) Basic Research Project No. JCYJ20120618115257259.

References

[1] H. K. Aghajan and A. Cavallaro, *Multi-Camera Networks: Principles and Applications*, ser. Academic Press. Academic Press, 2009.

[2] R. Vaisenberg, S. Mehrotra, and D. Ramanan, "SEMARTCam Scheduler: Semantics driven real-time data collection from indoor camera networks to maximize event detection," *Journal of Real-Time Image Processing (JRTIP)*, vol. 5, no. 4, pp. 215–230, December 2010.

[3] V. Isler and R. Bajcsy, "The sensor selection problem for bounded uncertainty sensing models," *IEEE Transactions on Automation Science and Engineering*, vol. 3, no. 4, pp. 372–381, 2006.

[4] J. Williams, J. Fisher, and A. Willsky, "Approximate dynamic programming for communication-constrained sensor network management," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4300–4311, 2007.

[5] A. Cenedese, F. Cerruti, M. Fabbro, C. Masiero, and L. Schenato, "Decentralized task assignment in camera networks," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 126 –131.

[6] D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn, "Popular matchings," *SIAM Journal on Computing*, vol. 37, no. 4, pp. 1030–1045, 2007.

[7] C. Sng and D. Manlove, "Popular matchings in the weighted capacitated house allocation problem," *Journal of Discrete Algorithms*, vol. 8, no. 2, pp. 102–116, 2010.

[8] H. Gabow, "An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems," in *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. ACM, 1983, pp. 448–456.

[9] R. W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. E. Paluch, "Rank-maximal matchings," *ACM Transactions on Algorithms (TALG)*, vol. 2, no. 4, pp. 602–610, Oct. 2006.

[10] T. Kavitha and C. D. Shah, "Efficient algorithms for weighted rank-maximal matchings and related problems," in *Proceedings of the 17th international conference on Algorithms and Computation*, ser. ISAAC'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 153–162.