Parallel computational optimization in operations research: A new integrative framework, literature review and research directions☆

Guido Schryen

Department of Management Information Systems, Paderborn University

Warburger Str. 100, 33098 Paderborn, Germany

Abstract

Solving optimization problems with parallel algorithms has a long tradition in OR. Its future relevance for solving hard optimization problems in many fields, including finance, logistics, production and design, is leveraged through the increasing availability of powerful computing capabilities. Acknowledging the existence of several literature reviews on parallel optimization, we did not find reviews that cover the most recent literature on the parallelization of both exact and (meta)heuristic methods. However, in the past decade substantial advancements in parallel computing capabilities have been achieved and used by OR scholars so that an overview of modern parallel optimization in OR that accounts for these advancements is beneficial. Another issue from previous reviews results from their adoption of different foci so that concepts used to describe and structure prior literature differ. This heterogeneity is accompanied by a lack of unifying frameworks for parallel optimization across methodologies, application fields and problems, and it has finally led to an overall fragmented picture of what has been achieved and still needs to be done in parallel optimization in OR. This review addresses the aforementioned issues with three contributions: First, we suggest a new integrative framework of parallel computational optimization across optimization problems, algorithms and application

☆Invited review

domains. The framework integrates the perspectives of algorithmic design and computational implementation of parallel optimization. Second, we apply the framework to synthesize prior research on parallel optimization in OR, focusing on computational studies published in the period 2008-2017. Finally, we suggest research directions for parallel optimization in OR.

Keywords: computing science, parallel optimization, computational optimization, literature review

1. Introduction

Parallel optimization has received attention in the operations research (OR) field already for decades. Drawing on algorithmic and computational parallelism in OR is appealing as real-life optimization problems in a broad range of application domains are usually NP-hard and even the implementation of (meta)heuristic optimization procedures may require substantial computing resources. It has been argued that parallelism is crucial to make at least some problem instances tractable in practice and to keep computation times at reasonable levels (Talbi, 2009; Crainic et al., 2006).[1] However, unsurprisingly, the application of parallel optimization has been hesitant because i) parallelizing algorithms is challenging in general from both the algorithmic and the computational perspective, and ii) a viable alternative to parallelizing algorithms has been the exploitation of ongoing increases of clock speed of single CPUs of modern microprocessors. But this growth process reached a limit already several years ago due to heat dissipation and energy consumption issues (Diaz et al., 2012). This development makes parallelization efforts (not only in optimization) much more important than it was in earlier times.

Fortunately, the need for parallelization has been acknowledged and accompanied by an increased availability of parallel computing resources. This availability is rooted in two phenomena: a) the rapid development of parallel hard-

[1] Impressive computational results of applying parallelization to the traveling salesman problem (TSP) are reported by Crainic et al. (2006, p.2).

ware architectures and infrastructures, including multi-core CPUs and GPUs, local high-speed networks and massive data storage, and of libraries and software frameworks for parallel programming (Talbi, 2009; Crainic et al., 2006; Brodtkorb et al., 2013); b) the increased availability of parallel computing resources as commodity good to researchers, who have (free or low-priced) access to multi-core laptops and workstations, and even to high-performance clusters offered by universities and public cloud providers.

The benefits of exploiting parallel processing for optimization algorithms are multi-faceted. Searching the solution space can be speeded up for both exact and (meta)heuristic algorithms so that the optimal solution or a given aspiration level of solution quality, respectively, can be achieved quicker. Implementations can also benefit from improved quality of the obtained solutions, improved robustness, and solvability of large-scale problems (Talbi, 2009, p. 460f).

We found many published reviews on parallel optimization for particular problems, methodologies, applications, research disciplines, and technologies. Reviews of parallelization for particular optimization problems were provided for one-dimensional integer knapsack problems (Gerasch and Wang, 1994), vehicle routing problems (VRPs) (Crainic, 2008), non-linear optimization (Lootsma and Ragsdell, 1988), mixed integer programming (Nwana and Mitra, 2000) and multiobjective optimization (Nebro et al., 2005). Most of the reviews that we found focus on parallel optimization regarding particular methodologies. While branch-and-bound algorithms have been reviewed by Gendron and Crainic (1994), the majority of methodological literature reviews have focused on metaheuristics: reviews have addressed tabu search (TS) (Crainic et al., 2005), simulated annealing (SA)(Aydin and Yigit, 2005), variable neighborhood search (VNS) (Pérez et al., 2005), Greedy Randomized Adaptive Search Procedures (GRASPs) (Resende and Ribeiro, 2005), swarm intelligence algorithms (Tan and Ding, 2016), particle swarm optimization algorithms (Zhang et al., 2015), and different types of evolutionary algorithms, including genetic algorithms (GAs) (Adamidis, 1994; Luque et al., 2005; Cantú-Paz, 1998; Alba and Troya, 1999; Adamidis, 1994; Knysh and Kureichik, 2010), ant colony optimization algorithms (Pedemonte et al., 2011; Janson et al., 2005), scatter search (López et al., 2005) and evolutionary strategies (Rudolph, 2005). Several reviews have covered sets of metaheuristics (Cung et al., 2002; Alba et al., 2005; Crainic and Hail, 2005; Pardalos et al., 1995; Crainic and Toulouse, 2003, 2010; Crainic et al., 2014; Crainic, 2018, 2019; Alba et al., 2013) and hybrid metaheuristics (Cotta et al., 2005; Luna et al., 2005). Application- and discipline-oriented reviews of parallel optimization have been provided for routing problems in logistics (Schulz et al., 2013) and for parallel metaheuristics in the fields of telecommunications and bioinformatics (Nesmachnow et al., 2005; Trelles and Rodriguez, 2005; Martins and Ribeiro, 2006). Reviews that focus on particular parallelization technologies (in particular, General Purpose Computation on Graphics Processing Unit (GPGPU)) have been proposed by Boyer and El Baz (2013), Tan and Ding (2016) and Schulz et al. (2013).

We acknowledge the excellent work provided in these reviews, from which our review has benefited substantially. At the same time, we see several arguments that call for a new literature review. First, we did not find reviews that cover the most recent literature on the parallelization of both exact and (meta)heuristic methods published in the decade 2008-2017. During this time, substantial advancements in parallel computing capabilities and infrastructures have been achieved and used by many OR scholars so that an overview of modern parallel optimization in OR that accounts for these advancements when synthesizing and classifying the literature is beneficial. Second, based on different foci adopted in previous literature reviews, the concepts used to describe and structure prior literature differ. This heterogeneity is accompanied by a lack of unifying frameworks for describing parallel optimization across methodologies, application fields, and problems. This has led finally to an overall fragmented picture of what has been achieved and what still needs to be done in parallel optimization in OR. As a side effect, the heterogeneity with which parallelization studies in OR have been described in terms of algorithmic parallelization, computational parallelization and performance of parallelization is high, which is beneficial from a diversity perspective but also raises problems: First, it remains unclear for authors what should be reported in an OR study that draws on parallel optimization; second, our own experience based on screening and reading several hundreds of articles is that the heterogeneity makes it often time-consuming and in some case even impossible for readers to identify the aforementioned parallelization characteristics of a study, to classify the study accordingly and to compare studies with each other.

Accounting for the aforementioned challenges, we provide three contributions in this literature review. First and to our best knowledge, we suggest the first universally applicable framework for parallel optimization in OR, which can be used by researchers to systematically describe their parallelization studies and position these in the landscape of parallel optimization without requirements on the application domain touched, the problem addressed, the methodology parallelized or the technology applied. In particular, the suggested framework integrates both algorithmic design and computational implementation issues of parallel optimization, which are usually being addressed separately in the literature. Second, we apply the integrative framework to synthesize prior research on parallel optimization in the field of OR published in the decade 2008-2017, focusing on those studies which include computational experiments. Finally, we suggest research directions, including recommendations, for prospective studies on parallel optimization in OR.

We structure our review as follows: In Section 2, we develop a framework for computational studies on parallel optimization. In Section 3, we define the scope and literature selection process of our review, before we review the literature in Section 4 based on the suggested framework. We provide research directions for future research in Section 5 before we conclude our review in Section 6.

2. Parallelization Framework

Computational studies on parallel optimization usually report on four perspectives of parallelization (Gendron and Crainic, 1994; Alba and Luque, 2005; Crainic and Hail, 2005; Talbi, 2009; Pedemonte et al., 2011; Crainic, 2018, 2019): *object of parallelization*, *algorithmic parallelization*, *computational parallelization* and *performance of parallelization*. While our review of the literature revealed that most studies make either implicitly or explicitly use of the aforementioned perspectives, we also observed a high level of heterogeneity in terms of terminology, taxonomies of parallel algorithmic design, granularity of information on parallel implementation, and performance metrics used to report computational results. As a consequence, with an increasing body of computational studies, it has become challenging to gain an overview of computational achievements, to compare studies in terms of their achievements, to develop consistent taxonomies for computational studies, and to identify white spots that need further research.

In order to mitigate the aforementioned problems in the field of parallel optimization, we suggest a new descriptive framework of computational parallel optimization studies (see Figure 1). The scope of the applicability of the proposed framework in the area of parallel optimization is wide with regard to two dimensions: First, it does not make any assumptions about the addressed application domain, the optimization problem to solve, the parallelized methodology or the applied technology. We denote this broad applicability as *horizontal integration*, referring to the horizontal layers in Figure 1. Second, it integrates the aforementioned perspectives (layers) and is based on well-established principles in the literature on algorithmic and computational parallelization. Similarly, we refer to this broad applicability as *vertical integration*, which brings together the – usually separately applied – perspectives on parallel optimization found in the disciplines of OR and computer science. In this context, our framework adopts an integrated view on parallel optimization.

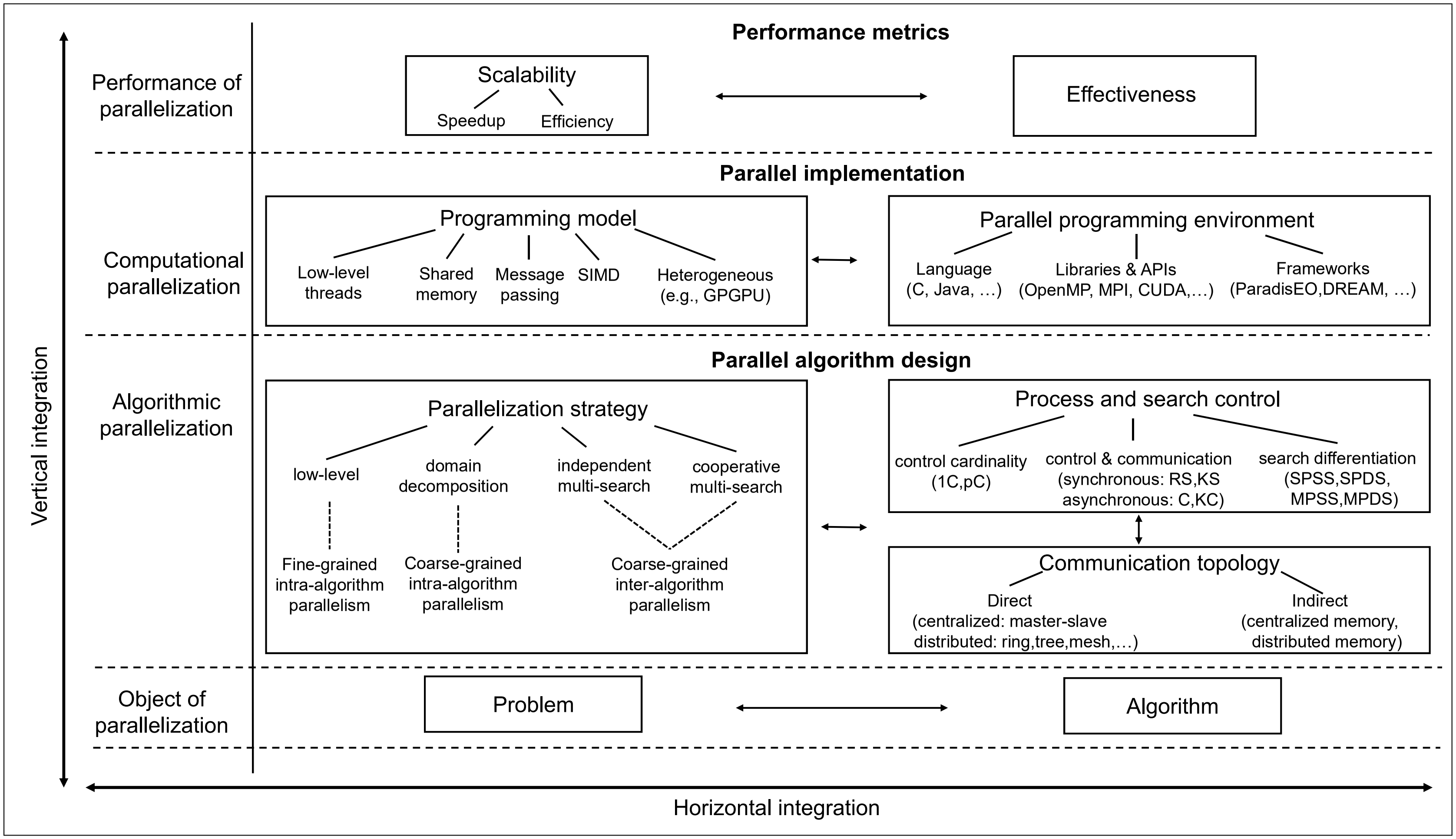

Figure 1: Integrative framework for computational parallelization studies in OR

2.1. Object of parallelization

The object of parallelization comprises the OR problem to be solved (e.g., TSP, VRP, JSSP) and the algorithm to be applied (e.g., b&b, GA, SA, TS), which effect each other. Problem types and algorithm types are both described in detail in Section 4.2.

2.2. Algorithmic parallelization

The algorithmic parallelization refers to the methodological perspective on how parallelism is applied to solve an optimization problem by decomposition. As suggested for metaheuristics (Crainic, 2019) , we detail this perspective by distinguishing various types of *parallelization strategy*, *process and search control*, and *communication topology* (see Figure 1). Parallelization strategies have been defined according to the source of parallelism (Cung et al., 2002; Crainic and Toulouse, 2003; Crainic and Hail, 2005; Crainic and Toulouse, 2010; Crainic, 2019). Four types are distinguished: (1) Functional parallelism applies when decomposition occurs at the algorithm level by, for example, evaluating neighbor solutions or computing the fitness of a solution in parallel. This parallelization strategy does not alter the algorithmic logic, the search space or the behavior of the sequential version, and it is thus also referred to as *low-level*. As parallelism occurs at a low level inside a single algorithm, we coin the term *fine-grained intra-algorithm parallelism*. Since the overall search follows only a single search path, this type of parallelism has also been denoted as *single-walk parallelization*, in contrast to the following strategies, where the overall search follows multiple trajectories and are referred to as *multiple-walk parallelization strategies* (Cung et al., 2002). (2) *Domain decomposition* refers to the approach of separating and exploring the search space explicitly yielding a number of smaller and easier to solve subproblems to be addressed simultaneously by applying the same sequential algorithm. The partial solutions are finally used to reconstruct an entire solution of the original problem. The separation of the search space may be obtained, for example, by discarding or fixing variables and constraints. This separation may result in a partition (disjoint subsets) or a coverage (subsets may overlap) of the overall search space. In contrast to the low-level strategy, where parallelism occurs at a local and predefined part of the algorithm, domain decomposition involves concurrent explorations of subspaces using the same algorithm. Thus, we introduce the term *coarse-grained intra-algorithm parallelism*. (3) Separating the search space can also be performed implicitly through concurrent explorations of the search space by different or differently parameterized methods. When the concurrent execution of methods does not involve any exchange of information prior to identifying the best overall solution at the final synchronization step, the parallelization strategy is referred to as *independent multi-search*, which can be perceived as *coarse-grained inter-algorithm parallelism*. (4) When the concurrent execution of methods and their explorations of subspaces involves the exchange of information through cooperation mechanisms while the search process is in progress, *cooperative multi-search* occurs. The sharing of information may even be accompanied with the creation of new information out of exchanged data. As the interactions of the cooperative search algorithms specify the global search behavior, a new metaheuristic in its own right emerges (Crainic and Toulouse, 2008). While cooperation yields in many cases a collective output with better solutions than a parallel independent search (Crainic, 2019), exchanges should not be too frequent to avoid communication overheads and premature "convergence" to local optima (Toulouse et al., 2000, 2004). As in the case of independent multi-search, also cooperative multi-search can be seen as coarse-grained inter-algorithm parallelism. Finally, it should be noticed that parallelization strategies are not mutually incompatible and may be combined into comprehensive algorithmic designs (Crainic et al., 2006; Crainic, 2019). For example, low-level and decomposition parallelism have been jointly applied to branch-and-bound (Adel et al., 2016) and dynamic programming (Vu and Derbel, 2016), (Maleki et al., 2016), and low-level parallelism and cooperative multi-search have been applied to a hybrid metaheuristic (Munawar et al., 2009) which uses a genetic algorithm and hill climbing.

While the aforementioned parallelization strategies have been formulated for the class of metaheuristics, the strategy-defining principles are of general nature of parallelizing optimization algorithms so that the scope of applicability of the parallelization strategies can be straightforward extended to other algorithm classes, including exact methods and (problem-specific) heuristics. For example, Gendron and Crainic (1994) have defined three types of parallelism for branch-and-bound: their type 1 parallelism refers to parallelism when performing operations on generated subproblems, such as executing the bounding operation in parallel for each subproblem. This type can be perceived as low-level parallelism. Parallelism of type 2 consists of building the branch-and-bound tree in parallel by performing operations on several subproblems concurrently. This type of parallelism involves an explicit separation of the search space and can, thus, be perceived as domain decomposition. Finally, the case of type 3 parallelism implies that several branch-and-bound trees are built in parallel, with the trees being characterized by different operations (branching, bounding, testing for elimination, or selection). This parallelism includes the option to use the information generated during the construction of a tree for the construction of another one. When such information is exchanged, type 3 parallelism can be perceived as cooperative multi-search, otherwise it corresponds to independent multi-search. The straightforward matching of parallelization strategies for metaheuristics with types of parallelism defined for an exact method supports our previous argument that the four parallelization strategies can be applied to the general "universe" of optimization algorithms.

Process and search control refers to how the global problems-solving process is controlled, how concurrent processes communicate with each other, and how diverse the overall search process is. We adopt the three dimensions suggested by Crainic and Hail (2005): *Search control cardinality* determines whether the global search is controlled by a single process (1-control, 1C)) or by several processes (p-control, pC) which may collaborate or not. *Search control and communications* refers to how information is exchanged between processes and distinguishes between synchronous and asynchronous communication. In the former case, all concerned processes have to stop and engage in some form of communication and information exchange at specified moments (e.g., number of iterations) exogenously determined. In the latter case, processes are in charge of their own search as well as of establishing communications with other processes, and the global search terminates once each individual search stops. Both synchronous and asynchronous communication can be further qualified with regard to whether additional knowledge is derived from communication, leading to four categories of control and communication: rigid (RS) and knowledge synchronization (KS) in the synchronous case, and collegial (C) and knowledge collegial (KC) in the asynchronous case. Finally, the diversity of search may vary according to whether concurrently executed methods start from the same or different solutions, and to whether their search follows the same or different logics[2]; the diversity of search is also referred to as *search differentiation.* From these two dimensions the following four classes can be derived: 1. *same initial point/population, same search strategy (SPSS)*; 2.*same initial point/population, different search strategies (SPDS)*; 3. *multiple initial points/populations, same search strategies (MPSS)*; 4. *multiple initial points/populations, different search strategies (MPDS).* While the term "point" relates to single-solution methods, the notion "population" is used for population-based ones, such as genetic algorithms or ant colony optimizations. As in the case of parallelization strategies described above, the three dimensions of process and search control have been suggested for the classification of metaheuristics (Crainic and Hail, 2005; Crainic, 2018, 2019) but can be extended straightforward to other classes of optimization algorithms.

When concurrent processes exchange information, they may communicate with each other in a direct or indirect way. Direct communication involves message-based communication along some communication topology, such as a tree, ring, or fully connected mesh (Talbi, 2009; Crainic, 2019). This communication topology needs to be projected on a physical interconnection topology as part of the implementation design. In contrast, indirect communication involves

[2]Two logics are characterized as "different" even when based on the same methodology (e.g., two tabu searches or genetic algorithms) if they vary in terms of components (e.g., neighborhoods or selection mechanism) or parameter values (Crainic, 2019).

the use of a centralized or distributed memory, which are used as shared data resources of concurrent processes (Crainic, 2019).

The three perspectives of parallel algorithm design, namely parallelization strategy, process and search control, and communication topology, are linked together (Crainic, 2018, 2019). Low-level parallelization is generally targeted in 1C/RS/SPSS designs, with the 1C (control cardinality) being implemented with a master-slave approach. Examples are the neighborhood evaluation of a local search heuristic, and the application of operators and the determination of fitness values in a GA. Domain decomposition is often implemented using a master-slave 1C/RS scheme with MPSS or MPDS search differentiation but can also be performed in a pC, collegial decision making framework with MPSS or MPDS search differentiation. Independent multi-search is inherently a pC parallelization strategy, which follows from the same or different starting point(s)/population(s) with or without different search strategies (i.e., SPDS, MPSS or MPDS search differentiation). As the concurrently executed search processes do not exchange information prior to the final step, they follow the RS control and communication paradigm. Finally, cooperative multi-search is also a pC parallelization strategy, which may start from possibly different starting points/populations and may follow different search strategies (i.e., SPDS, MPSS or MPDS search differentiation). In contrast to independent multi-search, information is exchanged between processes during the search. This exchange of information can vary in different ways, which results in a large diversity of cooperation mechanisms. First, different types of information may be exchanged, including "good" solutions and context information. Second, cooperating processes may exchange information directly by sending messages to each other based on a given communication topology, or indirectly using memories which act as data pools shared by processes. A third option distinguishes between synchronous and asynchronous cooperation, where processes either need to stop its activities' until all others are ready or not, respectively.

2.3. Computational parallelization

When parallel algorithms are implemented and executed in modern computational environments, different parallel programming models may be applied in a variety of programming environments. Albeit being intertwined (see, for example, (Talbi, 2009)), they represent different facets of parallel implementation from a conceptual perspective. Four (pure) parallel programming models can be distinguished: threads, shared memory, message passing (Diaz et al., 2012; Talbi, 2009) and single-instruction-multiple-data (SIMD). In the thread programming model, lightweight processes (threads) are executed, where the communication between threads is based on shared addresses. The shared memory programming model, where, too, tasks share a common address space, operates at a higher abstraction level than threads. Today, both the thread and the shared memory model are executed on a multi-core CPU architecture on a single computer node. In contrast, in the message passing programming model the communication between processes is done by sending and receiving messages. Each process has its own address space that is not shared with other processes. This model is designed for execution in computer clusters, where different nodes are connected through high-speed networks. Note that, depending on the particular parallel programming model, parallel executed software parts are labeled differently usually as *threads*, *tasks* or *processes*. Finally, SIMD exploits data parallelism by operating a single instruction on multiple data on a vector processor or array processor. Beyond the pure parallel programming models sketched above, the heterogeneous model *General Purpose Computation on Graphics Processing Unit (GPGPU)* has received increasing attention (e.g., (Brodtkorb et al., 2013)). GPGPU harnesses the capabilities of multi-core CPUs and many-core GPUs, where threads are executed in parallel on GPU cores and where GPUs can have different levels of shared memory; in this sense, we can speak of heterogeneous systems (Diaz et al., 2012). Other heterogeneous models are distributed shared memory models and field programmable gate arrays (FPGAs). In modern computing environments, (pure or heterogeneous) parallel programming models are sometimes combined with each other by, e.g., jointly using threads and GPGPU, shared memory and message passing, or threads and message passing (Diaz et al., 2012). Such approaches are referred to as *hybrid models*.

Parallel programming environments are related to parallel programming models and comprise languages, libraries, APIs (application programming interfaces) and frameworks.

2.4. Parallel performance metrics

The general purpose of parallel computation is to take advantage of increased processing power to solve problems faster or to achieve better solutions. The former goal is a matter of *scalability*, which is defined as the degree to which it is capable of efficiently utilizing increased computing resources. Performance measures of scalability fall into two main groups: *speedup* and *efficiency*. Speedup $S_p := \frac{S}{T_P}$ is defined as the ratio of sequential computation time S to parallel computation time T_p when the parallel algorithm is executed on p processing units (e.g., cores in a multicore processor architecture). The serial time S can be measured differently, leading to different interpretations of speedup (Barr and Hickman, 1993): When S refers to the fastest serial time on any serial computer, speedup is denoted as *absolute*. Alternatively, S may also refer to the time required to solve a problem with the parallel code on one processor. This type of speedup is qualified as *relative*. When real-time reduction is considered as the primary objective of parallel processing, absolute speedup is the relevant type. While speedup relates serial to parallel times, efficiency $E_p := \frac{S_p}{p}$ relates speedup to the number of processing units used. With the definition of efficiency, we can qualify speedup as *sublinear speedup* ($E_p < 1$), *linear speedup* ($E_p = 1$), or *superlinear speedup* ($E_p > 1$). Sublinear speedup is often due to serial parts of a parallel algorithm and several reasons for a nonvanishing serial part can be distinguished. Superlinear speedup can occur, for example, when during the parallel execution of a branch-and-bound algorithm one processor finds a good bound early in the solution process and communicates it to other processors for truncation of their search domains (Barr and Hickman, 1993). Finally, it should be noticed that while the application of speedup and related efficiency concepts to algorithms which have a "natural" serial version is straightforward, their unmodified application to multi-search algorithms, which are parallel in nature, does not make much sense as no basis of comparison is available.

A second important performance measure in parallel optimization is the solution quality achieved through parallelization. Solution quality can be measured in various ways. When the optimal solution value or a bound of it is known, the relative gap to (the bound of) the optimal value can be determined. A second option is to relate the achieved solution quality with that obtained from sequential versions of the parallelized algorithm (relative improvement). However, this option requires that a sequential version of the parallel algorithm exists in terms of unchanged algorithmic logic and the trajectory through the search space. This is not the case, for example, when cooperative multi-search occurs, which defines a new algorithm due to cooperation. Finally, the solution quality obtained through parallelization may be compared with the quality of the best known solution obtained from any serial implementation (absolute improvement). Overall, the goal of achieving better solutions can be perceived as an issue of *effectiveness*.

3. Scope and literature selection process

The focus of our literature review lies on computational studies of parallel optimization, where physical or virtual parallel computing architectures have been applied to OR problems, such as TSPs, VRPs and FSSPs (flow shop scheduling problems). Due to the interdisciplinary nature of the OR field, such studies are not only found in OR outlets but also in those of many other disciplines, including management science, mathematics, engineering, natural sciences, combinations of engineering and natural sciences (such as chemical engineering), computer science, bioinformatics, material science, geology and medicine. While we include outlets of these disciplines in our search (see the succeeding subsection), we would like to stress that the focus of our review lies on studies on OR problems and that it is beyond the scope of this review to identify and classify all articles of parallel optimization addressing problems in related fields or even across all fields (optimization in general). Adopting this view, we exclude from our review, for example, mathematical studies on parallelizing matrix computations or on conjugate gradient methods, computer science studies on load balancing issues in parallel computing environments or on solving hard problems in theoretical computer science (e.g., the subset sum problem), and parallel optimization studies across fields, such as those addressing the effects of migration in parallel evolutionary algorithms. We also exclude works on parallel optimization when their purpose lies in designing or implementing other methodologies, such as simulation, data analysis, data mining, machine learning and artificial intelligence. We further exclude meta optimization (calibrating parameters of optimization models or methodologies). We explicitly acknowledge the importance of these areas but they deserve and need dedicated literature reviews. Finally, from a technological perspective, we also do not consider distributed optimization that makes use of geographically dispersed computers and allows using grids, which comprise networks of many, often thousands or even millions of single computers. This field applies programming models and parallel programming environments that differ from those used in our framework, and it would need a dedicated literature review, too.

Accounting for the previously described scope of our review, we implemented different streams of literature search. A detailed description of the literature search process is provided in the online Appendix A. Although having implemented different streams of search, we admit that the application of our search procedure does not guarantee to identify all computational studies of parallel optimization in OR and that we may have overlooked studies. However, we are confident to have acquired a body of literature that is sufficiently comprehensive to draw a firm picture of computational parallelization in OR during the decade 2008-2017.

4. Literature survey

In this section, we provide a synthesis of the literature published in the decade 2008-2017. We first offer a brief meta analysis, then we analyze the body of literature with regard to which optimization problems have been solved by which (parallelized) algorithms before we present the findings of our literature analysis, structured along optimization algorithms and based upon the framework suggested above. Findings on (i) effectiveness and (ii) parallel programming environments are not presented here because (i) effectiveness results have been reported only rarely and in partially inconsistent ways in the studies of our sample, making comparisons of results difficult, and (ii) parallel programming environments should be considered across algorithms. We discuss both topics in Section 5. With regard to speedup, we qualify it by efficiency when reported in a study. When GPGPU is used as programming model, we only report speedup values without providing the number of parallel processing units or information on efficiency. The reason is that the number of parallel working units (usually GPGPU threads) needs to be interpreted different from that counting other parallel working units (CPU threads, processes) so that efficiency usually being defined as the ratio of speedup and the number of parallel processing units is not applied here. Details on this issue as well as the coding of all studies in our sample are provided in the online Appendix B.

4.1. Meta analysis

Overall, our sample consists of 206 studies, with 164 studies published in 77 different journals, 38 studies published at 36 different workshops, symposiums, conferences or congresses, and four studies published as book chapters. The joint distribution of articles over scientific outlets and years is summarized in Table 1, which shows that (1) there is no clear temporal development of the numbers of papers published per year, (2) while the number of scholarly outlets (journals, proceedings, etc.) which have published computational studies on parallel optimization in OR is high, only nine outlets have published at

Outlet	Year										Sum
	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	
ASC				2	1		1	1	1	2	8
CIE			1			1		1	1	1	5
COR	1		1		3	2	2			2	11
CCPE						1		1		3	5
EJOR		3	1		1	1	1	3	2	2	13
IJOC	1	1		2					1	1	6
JPDC	1	1	1	2		4	1	1	1		12
JSC				1			1		1	2	5
PC	1			1	1			2	2	1	8
Other journals	7	5	5	12	12	11	7	10	12	13	91
Proceedings	3	5	7	11	4	4	3		1		37
Book chapters	2	1		1	1						5
Sum	15	15	16	32	22	24	16	18	21	27	206

ASC: Applied Soft Computing
CIE: Computers & Industrial Engineering
COR: Computers & Operations Research
CCPE: Concurrency and Computation-Practice & Experience
EJOR: European Journal of Operational Research
IJOC: INFORMS Journal on Computing
JPDC: Journal of Parallel and Distributed Computing
JSC: Journal of Supercomputing
PC: Parallel Computing

Table 1: Joint distribution of selected articles over scientific outlets and years

least five articles during the decade 2008-2017 and only three outlets (namely, *Computers & Operations Research*, *European Journal of Operational Research*, *Journal of Parallel and Distributed Computing*) have published more than ten articles in the same period. Overall, this publication landscape does not reveal clear clusters in terms of time or outlet, it rather shows that computational and parallel optimization in OR has been covered permanently (and) distributed over many outlets rooted in different yet related academic disciplines, including *OR*, *Computer Science* and *Engineering*. Apparently, this research area is of multidisciplinary relevance.

4.2. Problem types and parallelized algorithms

We now describe the identified body of literature from the perspective of problem types and types of parallelized algorithms. Table 2 shows the joint

Alg. type	Problem type													Sum
	AP	FLP	FSSP	GTP	JSSP	KP	BFP	MILP	MSP	SOP	TSP	VRP	Other	
B-a-X	1		7	3	2	3	2	4		2	3		13	40
DP				2		3				1			4	10
IPM										2			2	4
PSEA				2		1		1						4
PSH		1	1	1				1			2		6	12
TS	4		5		2		1		1		2	5	3	23
SA			2		2		1		1		1	3	1	11
VNS			1		2	1			1			4	2	11
GRAS													2	2
OSSH						1								1
GA	2	2	3	1	1		3		3		3		10	28
OEA		1				1	2		1		1	1	6	13
SSPR			1										1	2
ACO	2										12	2		16
PSO		1		2	1		3						5	12
BCO							2		1					3
FA							1							1
HM	1	1	2	1	1	1	2		3		4	2	7	25
OH				1								1	2	4
MH		1										1		2
MS	1											1	1	3
Sum	11	7	22	13	11	11	17	6	11	5	28	20	65	227

Optimization Problem Type	Algorithm type
AP: Assignment Problem	Exact algorithms:
FLP: Facility Location Problem	B-a-X: Branch & X
FSSP: Flow Shop Scheduling Problem	DP: Dynamic programming
GTP: Graph Theory Problem	IPM: Interior point method
JSSP: Job Shop Scheduling Problem	PSEA: Problem-specific exact algorithms
KP: Knapsack Problem	PSH: Problem-specific heuristics
BFP: Benchmark function optimization problem(s)	Single-solution based metaheuristics:
MILP: (Mixed) Integer Linear Program	TS: Tabu search
MSP: Machine Scheduling Problem	SA: Simulated annealing
SOP: Stochastic Optimization Problem	VNS: Variable neigborhood search
TSP: Traveling Salesman Problem	GRAS: (Greedy randomized) adaptive search
VRP: Vehicle Routing Problem	OSSH: Other single solution heuristics
	Population-based metaheuristics:
	GA: Genetic algorithm
	OEA: Other evolutionary algorithms
	SSPR: Scatter search & path relinking
	ACO: Ant colony optimization
	PSO: Particle swarm optimization
	BCO: Bee colony optimization
	FA: Fireworks algorithm
	HM: Hybrid metaheuristics
	OH: Other heuristics
	MH: Matheuristics
	MS: Multi-search algorithms

Table 2: Joint distribution of selected articles over problems and (parallelized) algorithms

distribution of articles over these two dimensions. We identified problem types by, firstly, coding for each article of our sample the covered problem(s) and, secondly, consolidating problems to problem types widely used in the OR literature[3] Overall, we identified nine "application-oriented" problem types (AP, FLP, FSSP, GTP, JSSP, KP, MSP, TSP, VRP) and three "mathematically-oriented" problem types (BFP, MILP, SOP).[4] Adopting this distinction leads to assigning a study that, for example, formulates a TSP as a mixed-integer linear program to the problem class "TSP" rather than to the class "MILP" as it is TSP instances that are focused and not MILP instances in general. Conversely, studies assigned to one of the classes BFP, MILP or SOP explicitly address the related mathematically-oriented problem type and are not necessarily linked to any specific application . We consolidated all problem types for which only very few computational parallelization studies have been published to the category "Other"[5].

With regard to types of algorithms, we draw on a taxonomy suggested by Talbi (2009), who distinguishes between *exact algorithms* (e.g., branch-and-bound), *problem-specific heuristics* (e.g., Lin-Kernighan heuristic for the TSP), *single-solution based metaheuristics* (e.g., tabu search), and *population-based metaheuristics* (e.g., genetic algorithms)[6]. We extend the taxonomy by adding some algorithm types: *hybrid metaheuristics* refer to an metaheuristic where parts of a (meta)heuristic A are embedded into a step of a (meta)heuristic B; *matheuristics* refer to the interoperation of metaheuristics and (exact) mathematical programming techniques; *multi-search algorithms* refers to the combi-

[3] An example of consolidation is grouping the "multi-depot VRP" and the "VRPs with time windows" to the problem type "VRP".

[4] While application-oriented problem types (e.g., TSP) usually lead to mathematical formulations which have an overall and coherent logic across the components (objective function, constraints, variables, etc.) of a model, "mathematically-oriented" problem types (e.g., MILP) have mathematical formulations where single components have to meet mathematical assumptions (e.g., binary variables, linear terms) without requiring the overall model to refer to a specific application concept.

[5] When an article studies several "other" problem types, we did not count the number of other problem types but coded it as a single appearance of an "other problem type".

[6] The authors also suggest the type *approximation algorithms*, which we do not use in this review.

nation of several independent search algorithms, which may collaborate or not. Finally, we provide *other heuristics* as a residual type for those (meta)heuristics which do not fit to any of the aforementioned algorithm types.

It should be noticed that the sums of addressed problem types and parallelized algorithm types shown in Table 2 do not equal the sample size for different reasons: (i) some articles in our sample apply more than one algorithm type to a single problem type and/or investigate more than one optimization problem type; (ii) a few articles do not clearly reveal (from our perspective) the targeted problem or the applied algorithm, or they do not parallelize any algorithm but only the evaluation of the objective function; due to these reasons, we excluded five articles from the presentation in Table 2. Overall, it should be kept in mind that each combination of addressed problem type and parallelized algorithm type is a "case" of a study, where a single study may have several cases. The perspective on optimization problems addressed in computational parallelization studies shows that a broad range of problem types have been covered. Beyond the 12 problem types highlighted, the residual class of other problem types includes 63 cases, in which computational parallelization has been applied to mostly different problem types. However, we also notice that a set of 12 problem types account for more than 70% of all cases, with a focus on the TSP, the FSSP and the VRP, which jointly account for more than 30% of all cases. Similar results are obtained from adopting the algorithmic perspective. While a broad range of exact algorithms and single-solution, population-based and hybrid metaheuristics have been parallelized, only a few algorithm types (branch-and-X (X=bound, cut, price, etc.), GAs, hybrid metaheuristics, TS) account for more than 50% of all cases, with branch-and-X accounting for about 18%. Jointly adopting the problem and algorithmic perspective, again, shows a large diversity but in this case no large clusters occur. Only four combinations (ant colony optimization applied to the TSP, branch-and-X applied to the FSSP, TS applied to the FSSP, TS applied to the VRP) have been covered in at least five cases, but these four combinations account for only about 13% of all cases.

In the remainder of this section, we present parallel computational optimiza-

Algorithm type	Computational studies
Exact algorithms:	
Branch & X	(Mezmaz et al., 2014; Chakroun et al., 2013b; Herrera et al., 2017; Taoka et al., 2008; Ponz-Tienda et al., 2017; Ismail et al., 2014; Paulavicius et al., 2011; Christou and Vassilaras, 2013; McCreesh and Prosser, 2015; Eckstein et al., 2015; Carvajal et al., 2014; Borisenko et al., 2017; Gmys et al., 2017; Liu and Kao, 2013; Bak et al., 2011; Gmys et al., 2016; Silva et al., 2015; Barreto and Bauer, 2010; Vu and Derbel, 2016; Chakroun and Melab, 2015; Paulavičius and Žilinskas, 2009; Posypkin and Sigal, 2008; Chakroun et al., 2013a; Aitzai and Boudhar, 2013; Ozden et al., 2017; Cauley et al., 2011; Xu et al., 2009; Aldasoro et al., 2017; Pages-Bernaus et al., 2015; Lubin et al., 2013; Adel et al., 2016; Borisenko et al., 2011; Boukedjar et al., 2012; Carneiro et al., 2011; Galea and Le Cun, 2011; Herrera et al., 2013; Sanjuan-Estrada et al., 2011)
Dynamic programming	(Dias et al., 2013; Aldasoro et al., 2015; Maleki et al., 2016; Tan et al., 2009; Stivala et al., 2010; Boyer et al., 2012; Boschetti et al., 2016; Kumar et al., 2011; Rashid et al., 2010; Tran, 2010)
Interior point method	(Huebner et al., 2017; Hong et al., 2010; Lubin et al., 2012; Lucka et al., 2008)
Problem-specific exact algorithms	(Li et al., 2015; Rossbory and Reisner, 2013; Kollias et al., 2014; Bozdağ et al., 2008))
Problem-specific heuristics	(Dobrian et al., 2011; Ozden et al., 2017; Ismail et al., 2011; Bożejko, 2009; Lancinskas et al., 2015; Koc and Mehrotra, 2017; Redondo et al., 2016; Hemmelmayr, 2015; Benedicic et al., 2014; Gomes et al., 2008; Baumelt et al., 2016; Luo et al., 2015).
Single-solution based metaheuristics:	
Tabu search	(Rudek, 2014; Jin et al., 2012; Bozejko et al., 2017; Hou et al., 2017; Bozejko et al., 2013; Czapinski and Barnes, 2011; James et al., 2009; Czapiński, 2013; Bukata et al., 2015; Cordeau and Maischberger, 2012; Wei et al., 2017; Janiak et al., 2008; Shylo et al., 2011; Jin et al., 2014; Bożejko et al., 2016; Jin et al., 2011; Maischberger and Cordeau, 2011; Van Luong et al., 2013; Dai et al., 2009; Melab et al., 2011)
Simulated annealing	(Thiruvady et al., 2016; Rudek, 2014; Defersha, 2015; Mu et al., 2016; Wang et al., 2015; Ferreiro et al., 2013; Lou and Reinitz, 2016; Banos et al., 2016; Bożejko et al., 2009, 2016; Lazarova and Borovska, 2008)
Variable neigborhood search	(Yazdani et al., 2010; Lei and Guo, 2015; Davidović and Crainic, 2012; Quan and Wu, 2017; Menendez et al., 2017; Eskandarpour et al., 2013; Coelho et al., 2016; Polat, 2017; Tu et al., 2017; Aydin and Sevkli, 2008; Polacek et al., 2008)
(Greedy randomized) adaptive search	(Caniou et al., 2012; Santos et al., 2010)
Other single solution heuristics	(Hifi et al., 2014)
Population-based metaheuristics:	
Genetic algorithm	(Massobrio et al., 2017; Liu et al., 2016; Dorronsoro et al., 2013; Defersha and Chen, 2008, 2010; Huang et al., 2012; Liu and Wang, 2015; Defersha and Chen, 2012; Homberger, 2008; Gao et al., 2009; Tosun et al., 2013; Zhang et al., 2016; Lu et al., 2014; Abu-lebdeh et al., 2016; Kang et al., 2016; He et al., 2010; Limmer and Fey, 2017; Abbasian and Mouhoub, 2013; Roberge et al., 2013; Lančinskas and Žilinskas, 2013; Lančinskas and Žilinskas, 2012; Lazarova and Borovska, 2008; Sancı and İşler, 2011; Umbarkar et al., 2014; Wang et al., 2012; Zhao et al., 2011; Vallada and Ruiz, 2009; Arellano-Verdejo et al., 2017)
Other evolutionary algorithms	(Fabris and Krohling, 2012; Pedroso et al., 2017; Cao et al., 2017; Dorronsoro et al., 2013; Aldinucci et al., 2016; Figueira et al., 2010; Derbel et al., 2014; Baños et al., 2014; Nebro and Durillo, 2010; Nowotniak and Kucharski, 2011; Redondo et al., 2008; Weber et al., 2011; Zhao et al., 2011; Izzo et al., 2009)
Scatter search & path relinking	(Kerkhove and Vanhoucke, 2017; Bożejko, 2009)
Ant colony optimization	(Ling et al., 2012; Cecilia et al., 2013; Delevacq et al., 2013; Zhou et al., 2017; Hadian et al., 2012; Yang et al., 2016; Cecilia et al., 2011; Skinderowicz, 2016; Abouelfarag et al., 2015; Lazarova and Borovska, 2008; You, 2009; Zhao et al., 2011; Yu et al., 2011b; Diego et al., 2012; Tsutsui, 2008; Dongdong et al., 2010)
Particle swarm optimization	(Aitzai and Boudhar, 2013; Yu et al., 2017; Roberge et al., 2013; Scheerlinck et al., 2012; Ze-Shu et al., 2017; Qu et al., 2017; Hung and Wang, 2012; Laguna-Sanchez et al., 2009; Mussi et al., 2011; Deep et al., 2010; Ding et al., 2013; Wang et al., 2008)
Bee colony optimization	(Luo et al., 2014; Davidovic et al., 2011; Subotic et al., 2011)
Fireworks algorithm	(Ding et al., 2013)
Hybrid metaheuristics	(Thiruvady et al., 2016; Delevacq et al., 2013; Arrondo et al., 2014; Patvardhan et al., 2016; Nesmachnow et al., 2012; Redondo et al., 2011; Mezmaz et al., 2011; Ku et al., 2011; Li et al., 2017; Yu et al., 2011a; Munawar et al., 2009; Ravetti et al., 2012; Ben Mabrouk et al., 2009; Subramanian et al., 2010; Scheerlinck et al., 2012; Czapinski, 2010; Banos et al., 2013; Olensek et al., 2011; Fujimoto and Tsutsui, 2011; Ibri et al., 2010; Lančinskas and Žilinskas, 2013; Van Luong et al., 2012; Xhafa and Duran, 2008; Zhao et al., 2011; Zhu and Curry, 2009)
Other heuristics	(Benedicic et al., 2014; Sathe et al., 2012; Juan et al., 2013; Sancı and İşler, 2011)
Matheuristics	(Stanojevic et al., 2015; Groer et al., 2011)
Multi-search algorithms	(Chaves-Gonzalez et al., 2011; Vidal et al., 2017; Lahrichi et al., 2015)

Table 3: Parallel computational optimization studies in OR

tion studies in OR grouped by algorithm types. An overview over the studies of our sample is given is Table 3.

Exact algorithms: The majority of studies that apply exact algorithms parallelize branch-and-X algorithms. These studies analyze a broad range of optimization problems. Almost all adopt domain decomposition as parallelization strategy using a 1C/C or pC/C scheme with MPSS search differentiation, and most studies which report on the used communication topology apply a (one- or multiple-tier) master-slave approach. These efforts are not surprising as they reflect a straightforward (and traditional) way to parallelize branch-and-X algorithms. In contrast, the landscape of applied parallel programming models is more diverse and includes approaches based on threads, message passing, shared memory and GPGPUs. With regard to the former three models, mostly sublinear or linear speedup has been reported but there are also a few studies (Ponz-Tienda et al., 2017; Borisenko et al., 2011; Galea and Le Cun, 2011) that report superlinear speedup. This speedup can be achieved, for example, when a parallel executed algorithm provides "good" bounds that allow pruning large parts of the search tree at early stages. The use of GPGPUs has shown mixed results in terms of speedup; however, in some cases the reported speedup is substantial (between 76.96 and 170.69) (Chakroun et al., 2013a), which makes GPGPUs highly appealing for parallelizing branch-and-X algorithms. However, it should also be acknowledged that several of these GPGPU studies have reported a high variance of speedup with regard to problem instances solved. Dynamic programming[7] is the second most often parallelized exact algorithm. Its parallelization in terms of addressed problems is quite diverse. In most cases, low-level is used as parallelization strategy with a 1C/RS scheme and SPSS search differentiation. The landscape of applied communication topologies is quite homogeneous, with almost all studies that report on the applied communication topology drawing on a (one- or multiple-tier) master-slave approach. In contrast, the set of implemented programming models is

[7] An introduction into parallel dynamic programming is provided by Almeida et al. (2006).

heterogeneous. Interestingly and in contrast to branch-and-X parallelization, the reported speedups are all sublinear. Studies that use GPGPUs report different ranges of speedup, with one study (Tran, 2010) reporting an exceptionally high speedup in the range of 900-2,500. In addition, we found only a few studies which parallelize the interior point method. All of these studies address stochastic optimization problems, using low-level parallelism in a 1C/RS scheme with SPSS search differentiation, and they achieve sublinear or linear speedup. While all studies apply message passing as parallel programming model, the topologies used differ. Finally, a few exact methods designed for specific optimization problems (the knapsack problem (Li et al., 2015), mixed integer linear programming (Rossbory and Reisner, 2013) and graph theory problems (Kollias et al., 2014; Bozdağ et al., 2008)) have been parallelized. While all four studies show sublinear or linear speedup, the characteristics of algorithmic and computational parallelization are different.

Single-solution based metaheuristics: Single-solution based metaheuristics manipulate and transform a single solution during the search. They can occur in many different forms and their parallelization has been discussed in (Melab et al., 2006; Talbi, 2009). Parallelization can occur at the solution level, iteration level and algorithmic level. While parallelizing at the solution and iteration level generally corresponds to low-level parallelization with a 1C/RS scheme and SPSS search differentiation, parallelization at the algorithmic level is open to the broad range of parallelization strategies, and process and search control options. Our literature review revealed that mainly three single-solution based metaheuristics have been parallelized: TS, SA and VNS. TS has been applied to a variety of optimization problems. Most studies apply parallelization at the solution or iteration level, thereby adopting low-level parallelization with a 1C/RS scheme and SPSS search differentiation and a master-slave communication topology. We found a few exceptions from this algorithmic parallelization pattern; for example, Jin et al. (2012); James et al. (2009); Jin et al. (2014, 2011) adopt cooperative multi-search parallelization of TS, and Dai et al. (2009) implement domain decomposition parallelization of TS. The landscape of applied parallel programming models is quite diverse and includes approaches based on threads, message passing, shared memory, SIMD, and GPGPUs. Speedup results are mixed, including superlinear speedup (Bozejko et al., 2013; Shylo et al., 2011). The implementation on GPGPUs has shown substantial differences with regard to speedup, reaching values up to 420 (Czapiński, 2013). The landscape of parallel SA studies, which have also been applied to a variety of optimization problems, is more diverse than that of GA studies. It has been addressed by all four parallelization strategies with varying types of process and search control and with different programming models. In contrast to this heterogeneity, most studies apply a master-slave communication topology. Only a few studies report the achieved speedup, which is mostly sublinear. We found one study (Ferreiro et al., 2013) that parallelizes SA using GPGPU and achieves speedups in the range of about 73.44-269.46. VNS has also been applied to many different problems with all four parallelization strategies and a variety of process and search control variations, communication topologies, and programming models. As in the case of SA, about half of the studies do not report on speedup and those which do report sublinear speedup, with the exception of Polacek et al. (2008), who achieve linear speedup. One study uses GPGPU (Coelho et al., 2016) and achieves a speedup in the range of 0.93-14.49. Additionally, we found two studies (Caniou et al., 2012; Santos et al., 2010) that parallelize (greedy randomized) adaptive search and one study (Hifi et al., 2014) that parallelizes large neighborhood search (subsumed under "other single solution heuristic (OSSH)" in Table 2).

Population-based metaheuristics: In contrast to single-solution based metaheuristics, in population-based algorithms a whole population of solutions is evolved. Most prominent classes of population-based metaheuristics include evolutionary algorithms, scatter search and path relinking, swarm intelligence algorithms, and bee colony optimization (Talbi, 2009). When population-based algorithms are parallelized, we distinguish three models which, albeit having been suggested originally for evolutionary algorithms in general and GAs in particular (Alba and Tomassini, 2002; Talbi, 2009; Agrawal and Mathew, 2004;

Melab et al., 2006; Cantú-Paz, 2005; Luque et al., 2005), can be applied to other classes of population-based algorithms as well: *global*, *island (with or without migration)*, and *cellular model*. In the global model, parallel techniques are used to speed up the operation of the algorithm without changing the basic operation of the sequential version. When the evaluation of the whole population is done in parallel, parallelism occurs at the iteration level; when the algorithm evaluates a single individuum in parallel, parallelism occurs at the solution level. In both cases, low-level parallelization applies. Island models typically run (identical or different) serial population-based algorithm on subpopulations to avoid getting stuck in local optima of the search space. If individuals can be transmitted between subpopulations, the island model is also referred to as *migration model*; however, island models can also occur without migration. While in the former case, migration usually leads to a cooperative multi-search, the latter case generally corresponds to independent multi-search parallelization. The cellular model may be seen as a special case of the island model where an island is composed of a single individual. It should be noted that the models may be applied jointly (Cantú-Paz (2005), for example, describes such model combinations for GAs).

Evolutionary algorithms belong to the types of algorithms that have attracted substantial parallelization efforts. A good overview of the diversity with which combinations of different parallelization strategies and programming models can be applied to evolutionary algorithms is provided by Limmer and Fey (2017). In our sample, we found a focus on GAs as a particular subclass of evolutionary algorithm; we subsume all evolutionary algorithms other than GAs under the residual subclass"other evolutionary algorithms". GAs have been parallelized for a variety of optimization problems. Most of the studies adopt the island model with migration (cooperative multi-search) with a pC/RS scheme and MPSS or MPDS search differentiation. Only a few studies use the island model without migration (independent multi-search) with a pC/RS scheme and MPSS search differentiation, or the global model (low-level) with a 1C/RS scheme and SPSS search differentiation. Interestingly, all but one study (Vallada and Ruiz, 2009) apply synchronous communication. In the presence of the island model, a diversity of communication topologies has been applied with mostly message passing being used as programming model. In contrast, when the global model is applied, threads or GPGPU are drawn upon and mostly the master-slave topology is implemented. The described correlation between the parallelization strategy and the parallel programming model is not surprising as the communication between (a usually moderate number of) islands through exchanging messages is appealing while the processing of (a usually large number of) individuals in a global population through (an often large number of) threads executed on a CPU or GPGPU seems appropriate. Only about half of the 27 GA studies that we found report speedup values. Speedup results are overall mixed, including superlinear speedup (Homberger, 2008; Abu-lebdeh et al., 2016). The application of GPGPUs has led to homogeneous results, with a maximum speedup of about 33 (Wang et al., 2012). Evolutionary algorithms other than GAs, such as differential evolution or immune algorithm, have been applied to a variety of optimization problems. Almost all of these studies adopt the island model with migration (cooperative multi-search) with a pC/RS scheme and MPSS or MPDS search differentiation. We found only two studies (Baños et al., 2014; Izzo et al., 2009) which report an asynchronous communication. We identified no pattern regarding the applied communication topology and programming model.

Swarm intelligence algorithms are inspired from the collective behavior of species such as ants, fish and birds. Subclasses of swarm intelligence algorithms for which we found parallelization studies are ant colony optimization (including ant colony systems and "MAX-MIN Ant Systems" (Dorigo and Stützle, 2004)), particle swarm optimization, and fireworks algorithms. Parallelization strategies of ant colony optimization can be classified according to the above mentioned three strategies of parallelizing population-based metaheuristics; i.e., global, island or cellular model. Here, we follow the suggestion of Randall and Lewis (2002) to distinguish the parallel evaluation of solution elements, parallel ant colonies (independent or interacting) and parallel ants. These strategies are specializations of the global model, island model (without or with migration), and cellular model, respectively, of population-based metaheuristics. Interestingly, most of the parallelization studies using ant colony optimization have addressed the TSP. VRPs (Yu et al., 2011b; Diego et al., 2012) and assignment problems (Tsutsui, 2008; Dongdong et al., 2010) have been solved by two studies each. Almost all studies use parallel ants or multiple ant colonies but, overall, the studies vary regarding parallelization strategies, process and search control, communication topologies and programming models. Those studies which qualify the achieved speedups, report sublinear speedups. The speedup achieved through GPGPU parallelization goes up to 25. Particle swarm optimization has been applied to solve a diverse set of optimization problems. Most of the parallelization studies make use of the global or island model, realized as low-level or cooperative multi-search parallelization, respectively, with a master-slave communication topology. The process and search control implementations differ, with only one study (Wang et al., 2008) reporting asynchronous communication. Mostly message passing and GPGPU are used as parallel programming model. Speedups achieved on GPGPU go up to about 190; studies not using the GPGPU model either do not report speedup values or show an overall diverse picture. In addition, we identified one study (Ding et al., 2013) that applies a fireworks algorithm.

Other population-based metaheuristics: We identified five studies that parallelize population-based metaheuristics other than evolutionary algorithms and swarm intelligence algorithms, namely scatter search and path relinking (Kerkhove and Vanhoucke, 2017; Bożejko, 2009), and bee colony optimization (Luo et al., 2014; Davidovic et al., 2011; Subotic et al., 2011). Addressed problems, algorithmic and computational parallelization characteristics as well as efficiency results (where reported) are quite diverse.

Hybrid metaheuristics: Hybrid metaheuristics are joint applications of several (meta)heuristics (Talbi, 2009; Crainic, 2019). They are "appropriate candidates" for the application of a(n) (independent or cooperative) multi-search strategy. A diverse set of optimization problems has been investigated with parallel hybrid metaheuristics. The combinations of (meta)heuristics include ant colony optimization and local search, GAs and local search, GAs and SA, and GAs and TS, among others. Due to the diverse set of combined (meta)heuristics, unsurprisingly, the studies differ substantially with regard to addressed problems, parallelization strategies, process and search and control, communication topologies and parallel programming models. Although none of these studies report a superlinear speedup, Zhu and Curry (2009) reports an achieved speedup of 403.91 when parallelizing a combination of ant colony optimization and pattern search with a GPGPU-based implementation.

Problem-specific heuristics, other heuristics, matheuristics, and multi-search algorithms: Problem-specific heuristics have been parallelized for a variety of optimization problems, including a graph theory problem (Dobrian et al., 2011), TSPs (Ozden et al., 2017; Ismail et al., 2011), a FSSP (Bożejko, 2009), a facility location problem (Lancinskas et al., 2015), a mixed integer linear program (Koc and Mehrotra, 2017), and several other problems (Redondo et al., 2016; Hemmelmayr, 2015; Benedicic et al., 2014; Gomes et al., 2008; Baumelt et al., 2016; Luo et al., 2015). We found four studies which parallelize heuristics that differ from all types described above: an agent-based heuristic (Benedicic et al., 2014), an auction-based heuristic (Sathe et al., 2012), a Monte Carlo simulation inside a heuristic-randomization process (Juan et al., 2013), and a random search algorithm (Sancı and İşler, 2011). We found two studies which parallelize matheuristics (Stanojevic et al., 2015; Groer et al., 2011) and three studies which suggest multi-search algorithms (Chaves-Gonzalez et al., 2011; Vidal et al., 2017; Lahrichi et al., 2015). Due to the diverse nature of the aforementioned studies, we do not look for patterns in algorithmic parallelization, computational parallelization and scalability results.

5. Research directions

Based on the analysis of the identified literature published in the covered period (2008-2017), we subsequently suggest some research directions which may help (re)focusing on those areas that did not get much attention or were even neglected during the focused period. We would like to note that the observation of the absence or rareness of certain types of studies primarily refers to the aforementioned period. Work published prior to this period and surveys published earlier than this review (see Section 1) have addressed some of the "white spots" in research identified for the aforementioned period, which calls for *re*-focusing on related research paths.

5.1. Publication landscape and overall prospective research

The analysis of publication data reveals that computational and parallel optimization in OR has been steadily attractive for many journals and conferences not only in the OR field but also in various neighbor disciplines. This broad interest is also reflected in the diverse landscape of which optimization problems have been solved by which (parallelized) algorithms. While this diversity shows the large relevance and broad applicability of computational parallelization in optimization, a closer look also reveals that the landscape is still fragmented despite the algorithmic accumulation of branch-and-X, GAs and TS studies and the problem accumulation of FSSPs, TSPs and VRPs. This makes it difficult to analyze which combinations of problems and algorithms are promising for parallelization and how the algorithmic and computational parallelization should be designed. It should be noted that in the presence of a broad scope of problems and algorithms in parallel optimization, the number of approximately 200 studies published in ten years is relatively low. Future research and education can benefit from fostering (knowledge on how to conduct) computational studies in parallel optimization to overcome the limitations imposed by fragmentation (recommendations 1a and 1b in Table 4).

5.2. Object of parallelization

From the algorithmic perspective, branch-and-X algorithms represent the largest cluster of computational parallelization studies. In a few studies, this parallelization has even led to superlinear speedup but in most cases "only" (sub)linear speedups have been achieved. Future research should shed more light on how to achieve superlinear speedups (recommendation 2a). With regard to dynamic programming, which is the second most often analyzed type of exact algorithms, the (sublinear) speedup achievements are less promising (see recommendation 2b). Again, our subsample of dynamic programming studies and their coding can serve as a basis for future investigations on more efficient dynamic programming parallelization, in particular on how to achieve superlinear speedup. We extend this recommendation to future research on parallelization of Lagrangean decomposition, which is – as dynamic programming – another methodology often used in the important field of stochastic optimization but which has hardly been parallelized. Parallelization efforts with regard to interior point methods are hardly existent, which asks for more research in this regard (recommendation 2c).

Among single-solution based metaheuristics, three metaheuristics have received particular attention regarding parallelization: TS, SA and VNS. For TS, speedup results are mixed, including two studies that report superlinear speedups, and the implementation on GPGPUs has shown substantial differences with regard to speedup. Future research should analyze this heterogeneous picture (recommendation 2d). With regard to SA and VNS, not much can be said on efficiency as, unfortunately, many studies do not report achieved speedups (see recommendation 2e). Beyond the aforementioned metaheuristics, other single-solution based metaheuristics, including *greedy randomized adaptive search*, *guided local search*, *fast local search*, and *iterated local search* (Gendreau et al., 2010, 2019), have not received much attention with regard to parallelization, which points to further research opportunities (recommendation 2f).

With regard to population-based metaheuristics, GAs are the most often parallelized type of algorithm. However, only a few studies provide speedup values, some of them reporting superlinear speedups. While these achievements are promising, not much knowledge about the factors that lead to superlinear speedup (see recommendation 2g) has been developed. Furthermore, parallelization results for GAs as well as other evolutionary algorithms are mainly based on synchronous communication so that not much is known about the potential of applying asynchronous communication (recommendation 2h) . The second and third most often parallelized type of population-based metaheuristics are ant colony optimization and particle swarm optimization, respectively. With regard to ant colony optimization, achieved speedups are not very promising and mostly limited to applications to the TSP. Regarding particle swarm optimization, speedup results are quite mixed, with a promising speedup value of about 190 reported when using the GPGPU model. These results show that further research on parallelizing ant colony optimization and particle swarm optimization is recommendable (recommendation 2i). Analogously to single-solution based metaheuristics, some algorithms of population-based methaheuristics, including SSPR, BCO and FA, have not received much attention, which shows avenues for further research (recommendation 2j).

Interestingly, we found only very few research on the parallelization of matheuristics. We believe that the parallelization of both of its' elements, metaheuristic components and exact mathematical programming techniques, are promising areas of future research (recommendation 2k).

Similarly few attention has been attracted by multi-search algorithms, which offer a straightforward parallelization approach through parallelizing the execution of independent search algorithms involved in multi-search. We consider this research stream, in particular cooperative multi-search algorithms, to be highly relevant for future research on parallelization (recommendation 2l).

Beyond the previously identified algorithmic research directions, future research should also adopt problem-specific perspectives (recommendation 2m).

5.3. Algorithmic parallelization and computational parallelization

The algorithmic parallelization in the studies of our sample has drawn on all four (pure) parallelization strategies and on combinations of pure strategies. Low-level parallelization is the most often implemented strategy, with 83 out of 206 studies having used this type of parallelism. The process and search control is usually a 1C/RS scheme with SPSS search differentiation. Most studies which use low-level parallelism apply a master-slave communication topology, which is a straightforward approach. However, there are several exceptions, including fully-connected meshs (e.g., (Huebner et al., 2017)) and trees (e.g., (Tan et al., 2009)). It would be useful to know under which conditions communication topologies other than the master-slave topology are advantageous for low-level parallelization (recommendation 3a). Interestingly, even for low-level parallelism a diverse set of parallel programming models and environments have been used, including message passing. This is a bit surprising as message passing is generally applied for the communication between "'heavy weight processes" executed on different computing nodes.

Domain decomposition as parallelization strategy occurs in 56 studies, with most of them parallelizing branch-and-X algorithms, which can be parallelized straightforward by decomposition. Regarding control cardinality, we found 1C and pC control modes applied similarly often. However, control and communication mostly follows an asynchronous, collegial scheme with no knowledge being exchanged between parallel processes; the used search differentiation is largely MPSS. Future research may explore opportunities that knowledge-based communication offer (recommendation 3b).

Independent multi-search as a parallelization strategy has been applied in only 18 studies, in contrast to cooperative multi-search, which has been implemented in 72 studies. This trend is encouraging as the potential of exchanging information between parallel processes in order to jointly achieve better solutions in less time has thereby been acknowledged by researchers. The vast majority of all studies which apply (independent or cooperative) multi-search uses a (synchronous) rigid synchronization (type "RS"); we identified only four studies (Groer et al., 2011; Bukata et al., 2015; Jin et al., 2014; Lahrichi et al., 2015) which make use of knowledge-based communication. Future research should foster the exploration of knowledge-based communication when multi-search is applied (recommendation 3c). - Parallelization strategies can be combined to exploit complementary ways of parallelizations. For example, low-level and domain decomposition parallelism have been jointly applied to branch-and-X algorithms (Vu and Derbel, 2016; Adel et al., 2016) and to dynamic programming (Maleki et al., 2016), and low-level and multi-search parallelism to genetic algorithms (Abbasian and Mouhoub, 2013; Munawar et al., 2009). In total, we found eight studies which apply such combinations. Future research should more intensively tap the potential that joint applications of different parallelization strategies offer (recommendation 3d). Finally, different parallelization strategies can be applied (separately) to the same algorithm and problem in order to compare their effectiveness and scalability and to determine most appropriate and inappropriate parallelizations. Although we identified as many as 21 studies which follow this path, we encourage scholars to intensify research in this regard (recommendation 3e).

A broad range of different communication topologies has been applied, with master-slave being the most often used topology. The appropriateness of a communication topology needs to be linked to the particular algorithm and the applied parallelization strategy so that no general recommendations are appropriate. However, in the sample of computational studies we found only a few studies (e.g., (Mezmaz et al., 2014; Herrera et al., 2013; Rashid et al., 2010; Aydin and Sevkli, 2008)) that have implemented more than one topology for one parallelization strategy of a particular algorithm. This low number calls for more studies that investigate multiple topologies for particular combinations of algorithms and parallelization strategies (recommendation 3f).

The parallel implementation of optimization algorithms has exploited overall a rich set of programming models and modern programming environments, including low-level threads (Java threads and POSIX threads), shared memory (mainly OpenMP), message passing (mainly MPI), and GPGPUs (mainly CUDA-based). In addition, also hybrid programming models, including message passing and shared memory, shared memory and GPGPU, threads and GPGPU, and message passing and threads, have been used in a few studies. Other programming models, such as SIMD, have only rarely been used. We found several studies which provide either no or incomplete information on the used parallel programming model(s). We recommend that studies report on the programming model and programming environment used for their parallelization (recommendation 3g).

Only a few studies report on their (re-)use of software frameworks for parallelization, such as ParadisEO (INRIA, n.d.) for parallel and distributed metaheuristics or Bob++ (Djerrah et al., 2006) for branch and bound parallelization. Reasons for not drawing on such frameworks can be manifold. Scholars may deliberately decide to not make use of them due to the inappropriateness of frameworks for their implementation case or due to too time-consuming efforts to get acquainted with the frameworks. Or, scholars are not aware of the existence of such frameworks. Either way, the development, propagation and use of re-usable software frameworks can substantially reduce the tedious and error-prone implementation of parallel optimization code (see recommendation 3h).

5.4. Performance of parallelization

Scalability is essential regarding the appropriateness of a parallel implementation of an optimization algorithm. Interestingly, in 70 out of 206 studies speedup values are not (completely) reported or speedup is interpreted different from how it is usually done (see Section 2.4); for example, some studies determine the speedup by executing the serial and the parallel code on different hardware, resulting in speedup values that are challenging to interpret. Other studies determine the speedup only of parts of an algorithm or use another parallel implementation as base (see Appendix B for more details). In such cases, speedup values are hardly comparable with those of other studies and, thus, limit the usefulness of scalability analysis (see recommendation 4a).

But even in case speedup is provided, comparisons with other studies need to be done carefully for several reasons: First, scalability results are difficult to compare with those of other studies when technological characteristics of parallel working units (or even of hardware environments) differ. For example, threads at the software level need to be distinguished from threads at the hardware level (hyperthreading), and MPI processes executed on different physical nodes may perform different from those executed on different cores on the same physical node. Second, values of weak speedup need to be distinguished from those of relative speedup (see Section 2.4). A list of issues related to speedup comparison is provided in Appendix B. We condense our suggestions in recommendation 4b.

We analyzed the studies in our sample with regard to how many parallel working units (threads or processes) have been used, which we refer to as *range of parallelization*. The number of parallel threads executed on a CPU has been mostly not above 32 and it reaches its maximum at 128. When message passing is used on one or several nodes, the number of parallel processes units has in most cases not exceeded 256 and it has reached its maximum at 8,192. Hybrid approaches mostly use up to 1,024 parallel units, with the maximum number having been 2,048. Overall, the range of parallelization is quite limited compared to the number of parallel units that are available in modern parallel computing environments (see recommendation 4c).

Our analysis of how studies in the literature have considered the effectiveness of parallelization (to obtain better solutions) showed that many studies do not analyze this category of performance and that those studies which provide effectiveness results use many different ways to report these. They apply different stop criteria (numbers of iterations, wall time, number of function evaluations, combinations of these criteria, etc.) and different evaluation criteria (objective value, relative gap to the best (known) solution value, numbers of instances solved to optimality, relative improvements, etc.), and often do not make the applied stop criteria explicit, which makes it difficult to assess parallel implementations and to compare studies with regard to effectiveness (see recommendation 4d).

5.5. Presentation of studies

Finally, having reviewed more than two hundreds of parallelization studies, we found that studies differ substantially in the way how information on parallelization is provided, to what extent information is made explicit, and in which section(s) of the paper which information on parallelization is provided. This heterogeneity may reflect different practices in various subfields and journals, and it not advisable to recommend any standardization in this regard. However, in several studies we found information on parallelization being reported incomplete, intransparent or distributed, which can make it tedious to fully understand the applied parallelization. The framework suggested in this paper may help to mitigate these issues when researchers adopt it and describe how it applies to their studies (recommendation 5).

6. Conclusion

This invited review suggests a new integrative framework for parallel computational optimization. It integrates the perspectives on parallel optimization found in the disciplines of OR and computer science, and it distinguishes four levels: i) object of parallelization, ii) algorithmic parallelization, iii) computational parallization, and iv) performance of parallelization. We apply this framework to synthesize the body of literature (206 studies published between 2008 and 2017) of parallel computational optimization in OR. It should be noticed that the applicability of the suggested framework is not limited to the OR field. Finally, we suggest several bundles of research recommendations for parallel computational optimization in OR, with the recommendations grouped along the layers of the suggested framework.

7. Acknowledgements

I am grateful for the support of the editor, Prof. Emanuele Borgonovo, and the many comments of the anonymous reviewers, who helped much in improving this review article. I acknowledge the great efforts and tedious work of Gerhard Rauchecker, Constanze Hilmer, Richard Schuster, Abdullah Burak, Melih Yilmaz, Henning Siemes and Philip Empl, who contributed much to the identification and the coding of many research papers. This research has been supported by a grant (no. 315925033) of the German Science Foundation (DFG).

Publication landscape and overall prospective research	
1a	Implementation of dedicated (tracks at) workshops and conferences and publication of edited books, such as (Alba, 2005; Talbi, 2006), and of special issues in journals
1b	Integration of parallel optimization and its application in modern parallel computing environments in curricula of OR education
Object of parallelization	
2a	Identification of those (algorithmic and computational) factors that drive superlinear speedup when parallelizing branch-and-X algorithms. The sample of 41 cases and their coding provided in this review offer a basis for this research.
2b	Identification of ways to make parallelization of dynamic programming and of Lagrangean decomposition more efficient and to achieve superlinear speedup. Our subsample of dynamic programming studies and their coding can serve as a basis for future investigations.
2c	Amplification of parallelization efforts with regard to interior point methods.
2d	Analysis of heterogeneous picture of efficiency of TS parallelization to identify those factors that are most promising.
2e	Amplification of scalability analysis with regard to parallelizations of SA and VNS.
2f	Extension of parallelization efforts to a more comprehensive set of single-solution based metaheuristics, including *greedy randomized adaptive search*, *guided local search*, *fast local search*, and *iterated local search*.
2g	Identification of those factors that drive superlinear speedup when parallelizing GAs.
2h	Application of asynchronous communication to genetic algorithms and other evolutionary algorithms.
2i	Amplification of parallelization efforts with regard to ant colony optimization and particle swarm optimization.
2j	Extension of parallelization efforts to a more comprehensive set of population-based metaheuristics, including *scatter search & path relinking*, *bee colony optimization*, and *fireworks algorithms*.
2k	Intensification of research on parallelizing matheuristics.
2l	Intensification of research on the parallelization of multi-search algorithms, in particular those which include collaboration.
2m	Adoption of problem-specific perspectives by analyzing which parallelization efforts (algorithms, parallel algorithm designs, parallel implementations) lead to which performance for a particular optimization problem. From Table 2 it can be seen that, in particular, FSSPs, TSPs, and VRPs have attracted fairly high number of parallelization studies that can be used for further analysis.
Algorithmic parallelization and computational parallelization	
3a	Identification of conditions under which communication topologies other than the master-slave topology are advantageous for low-level parallelization.
3b	Exploration of opportunities that knowledge-based communication offers in the case of domain decomposition.
3c	Exploration of knowledge-based communication when multi-search parallelism is applied.
3d	Tapping the potential that joint applications of different parallelization strategies offer.
3e	Comparisons of effects that different parallelization strategies have when applied to a particular algorithm and problem in order to determine (in)appropriate parallelization strategies in this case.
3f	Investigation of multiple strategies and/or multiple topologies for a particular algorithm in order to compare the performance of these alternatives.
3g	Documentation of programming model and programming environment used for parallelization.
3h	Development and propagation of easy-to-use and flexible software frameworks for parallel optimization.
Performance of parallelization	
4a	Provision of values of both speedup and efficiency with regard to serial implementations executed on the same hardware.
4b	Comparison of speedup and efficiency between algorithms of different studies needs to account for computational parallelization details and the type of speedup (e.g., relative or weak speedup) considered.
4c	Extension of the range of parallelization (in terms of parallel computing units) to analyze scalability at larger levels.
4d	Amplification of research on effectiveness of computational parallelization and documentation of applied stop and evaluation criteria.
Presentation of studies	
5	Application of frameworks for describing parallelization studies to avoid incompleteness, intransparency and distributed provision of parallelization information. The framework suggested in this paper may be used.

Table 4: Recommendations for future research on parallel computational optimization in OR

References

Abbasian, R., Mouhoub, M., 2013. A hierarchical parallel genetic approach for the graph coloring problem. Applied Intelligence 39, 510–528.

Abouelfarag, A.A., Aly, W.M., Elbialy, A.G., 2015. Performance analysis and tuning for parallelization of ant colony optimization by using OpenMP, in: Proceedings of the 14th IFIP TC 8 International Conference on Computer Information Systems and Industrial Management, pp. 73–85.

Abu-lebdeh, G., Chen, H., Ghanim, M., 2016. Improving performance of genetic algorithms for transportation systems: case of parallel genetic algorithms. Journal of Infrastructure Systems 22.

Adamidis, P., 1994. Review of parallel genetic algorithms bibliography. Aristotle Univ. Thessaloniki, Thessaloniki, Greece, Tech. Rep .

Adel, D., Bendjoudi, A., El Baz, D., Abdelhakim, A.Z., 2016. GPU-based two level parallel B&B for the blocking job shop scheduling problem. Applied Soft Computing , 747–755.

Agrawal, J., Mathew, T.V., 2004. Transit route network design using parallel genetic algorithm. Journal of Computing in Civil Engineering 18, 248–256.

Aitzai, A., Boudhar, M., 2013. Parallel branch-and-bound and parallel PSO algorithms for job shop scheduling problem with blocking. International Journal of Operational Research 16, 14–37.

Alba, E., 2005. Parallel metaheuristics: A new class of algorithms. volume 47. John Wiley & Sons.

Alba, E., Luque, G., 2005. Measuring the performance of parallel metaheuristics. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 2. pp. 43–62.

Alba, E., Luque, G., Nesmachnow, S., 2013. Parallel metaheuristics: recent advances and new trends. International Transactions in Operational Research 20, 1–48.

Alba, E., Talbi, E.G., Luque, G., Melab, N., 2005. Metaheuristics and parallelism. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 4. pp. 79–103.

Alba, E., Tomassini, M., 2002. Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation 6, 443–462.

Alba, E., Troya, J.M., 1999. A survey of parallel distributed genetic algorithms. Complexity 4, 31–52.

Aldasoro, U., Escudero, L.F., Merino, M., Monge, J.F., Perez, G., 2015. On parallelization of a stochastic dynamic programming algorithm for solving large-scale mixed 0-1 problems under uncertainty. Top 23, 703–742.

Aldasoro, U., Escudero, L.F., Merino, M., Perez, G., 2017. A parallel branch-and-fix coordination based matheuristic algorithm for solving large sized multistage stochastic mixed 0-1 problems. European Journal of Operational Research 258, 590–606.

Aldinucci, M., Campa, S., Danelutto, M., Kilpatrick, P., Torquati, M., 2016. Pool evolution: a parallel pattern for evolutionary and symbolic computing. International Journal of Parallel Programming 44, 531–551.

Almeida, F., González, D., Peláez, I., 2006. Parallel dynamic programming. John Wiley & Sons, Inc., Hoboken, New Jersey.. chapter 2. pp. 29–51.

Arellano-Verdejo, J., Godoy-Calderon, S., Alonso-Pecina, F., Guzman Arenas, A., Antonio Cruz-Chavez, M., 2017. A new efficient entropy population-merging parallel model for evolutionary algorithms. International Journal of Computational Intelligence Systems 10, 1186–1197.

Arrondo, A.G., Redondo, J.L., Fernandez, J., Ortigosa, P.M., 2014. Solving a leader-follower facility problem via parallel evolutionary approaches. Journal of Supercomputing 70, 600–611.

Aydin, M.E., Sevkli, M., 2008. Sequential and parallel variable neighborhood search algorithms for job shop scheduling, in: Metaheuristics for scheduling in industrial and manufacturing applications. Springer, pp. 125–144.

Aydin, M.E., Yigit, V., 2005. Parallel simulated annealing. John Wiley & Sons, Inc., Hoboken, New Jersey.. chapter 12. pp. 267–287.

Bak, S., Blazewicz, J., Pawlak, G., Plaza, M., Burke, E.K., Kendall, G., 2011. A parallel branch-and-bound approach to the rectangular guillotine strip cutting problem. INFORMS Journal on Computing 23, 15–25.

Baños, R., Ortega, J., Gil, C., 2014. Hybrid MPI/OpenMP parallel evolutionary algorithms for vehicle routing problems, in: Proceedings of the 18th International Conference on the Applications of Evolutionary Computation, Springer-Verlag Berlin. pp. 653–664.

Banos, R., Ortega, J., Gil, C., De Toro, F., Montoya, M.G., 2016. Analysis of OpenMP and MPI implementations of meta-heuristics for vehicle routing problems. Applied Soft Computing 43, 262–275.

Banos, R., Ortega, J., Gil, C., Fernandez, A., De Toro, F., 2013. A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. Expert Systems with Applications 40, 1696–1707.

Barr, R.S., Hickman, B.L., 1993. Reporting computational experiments with parallel algorithms: Issues, measures, and experts' opinions. ORSA Journal on Computing 5, 2–18.

Barreto, L., Bauer, M., 2010. Parallel branch and bound algorithm - a comparison between serial, OpenMP and MPI implementations. Journal of Physics: Conference Series 256, 012018.

Baumelt, Z., Dvorak, J., Sucha, P., Hanzalek, Z., 2016. A novel approach for nurse rerostering based on a parallel algorithm. European Journal of Operational Research 251, 624–639.

Ben Mabrouk, B., Hasni, H., Mahjoub, Z., 2009. On a parallel genetic-tabu search based algorithm for solving the graph colouring problem. European Journal of Operational Research 197, 1192–1201.

Benedicic, L., Stular, M., Korosec, P., 2014. A GPU-based parallel-agent optimization approach for the service coverage problem in UMTS networks. Computing and Informatics 33, 1025–1046.

Borisenko, A., Haidl, M., Gorlatch, S., 2017. A GPU parallelization of branch-and-bound for multiproduct batch plants optimization. Journal of Supercomputing 73, 639–651.

Borisenko, A., Kegel, P., Gorlatch, S., 2011. Optimal design of multi-product batch plants using a parallel branch-and-bound method, in: Proceedings of the 11th International Conference on Parallel Computing Technologies, Springer-Verlag Berlin. pp. 417–430.

Boschetti, M.A., Maniezzo, V., Strappaveccia, F., 2016. Using GPU computing for solving the two-dimensional guillotine cutting problem. INFORMS Journal on Computing 28, 540–552.

Boukedjar, A., Lalami, M.E., El-Baz, D., 2012. Parallel branch and bound on a CPU-GPU system, in: Proceedings of the 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 392–398.

Boyer, V., El Baz, D., 2013. Recent advances on GPU computing in operations research, in: Proceedings of the 27th International Conference on Parallel & Distributed Processing Symposium, pp. 1778–1787.

Boyer, V., El Baz, D., Elkihel, M., 2012. Solving knapsack problems on GPU. Computers & Operations Research 39, 42–47.

Bozdağ, D., Gebremedhin, A.H., Manne, F., Boman, E.G., Catalyurek, U.V., 2008. A framework for scalable greedy coloring on distributed-memory parallel computers. Journal of Parallel and Distributed Computing 68, 515–535.

Bożejko, W., 2009. Solving the flow shop problem by parallel programming. Journal of Parallel and Distributed Computing 69, 470–481.

Bozejko, W., Gnatowski, A., Pempera, J., Wodecki, M., 2017. Parallel tabu search for the cyclic job shop scheduling problem. Computers & Industrial Engineering 113, 512–524.

Bożejko, W., Pempera, J., Smutnicki, C., 2009. Parallel simulated annealing for the job shop scheduling problem, in: Proceedings of the 9th International Conference on Computational Science, Springer-Verlag Berlin. pp. 631–640.

Bozejko, W., Pempera, J., Smutnicki, C., 2013. Parallel tabu search algorithm for the hybrid flow shop problem. Computers & Industrial Engineering 65, 466–474.

Bożejko, W., Uchroński, M., Wodecki, M., 2016. Parallel metaheuristics for the cyclic flow shop scheduling problem. Computers & Industrial Engineering 95, 156–163.

Brodtkorb, A.R., Hagen, T.R., Schulz, C., Hasle, G., 2013. Gpu computing in discrete optimization. part i: Introduction to the gpu. EURO journal on transportation and logistics 2, 129–157.

Bukata, L., Sucha, P., Hanzalek, Z., 2015. Solving the resource constrained project scheduling problem using the parallel tabu search designed for the CUDA platform. Journal of Parallel and Distributed Computing 77, 58–68.

Caniou, Y., Diaz, D., Richoux, F., Codognet, P., Abreu, S., 2012. Performance analysis of parallel constraint-based local search, in: Proceedings of the 17th

ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, New York, NY, USA. pp. 337–338.

Cantú-Paz, E., 2005. Theory of parallel genetic algorithms. John Wiley & Sons, Inc., Hoboken, New Jersey.. chapter 18. pp. 423–445.

Cantú-Paz, E., 1998. A survey of parallel genetic algorithms. Calculateurs Paralleles, Reseaux Et Systems Repartis 10, 141–171.

Cao, B., Zhao, J., Lv, Z., Liu, X., 2017. A distributed parallel cooperative coevolutionary multiobjective evolutionary algorithm for large-scale optimization. IEEE Transactions on Industrial Informatics 13, 2030–2038.

Carneiro, T., Muritiba, A.E., Negreiros, M., de Campos, G.A.L., 2011. A new parallel schema for branch-and-bound algorithms using GPGPU, in: Proceedings of the 23rd International Symposium on Computer Architecture and High Performance Computing, pp. 41–47.

Carvajal, R., Ahmed, S., Nemhauser, G., Furman, K., Goel, V., Shao, Y., 2014. Using diversification, communication and parallelism to solve mixed-integer linear programs. Operations Research Letters 42, 186–189.

Cauley, S., Balakrishnan, V., Hu, Y.C., Koh, C.k., 2011. A parallel branch-and-cut approach for detailed placement. ACM Transactions on Design Automation of Electronic Systems 16.

Cecilia, J.M., Garcia, J.M., Nisbet, A., Amos, M., Ujaldon, M., 2013. Enhancing data parallelism for ant colony optimization on GPUs. Journal of Parallel and Distributed Computing 73, 42–51.

Cecilia, J.M., Garcia, J.M., Ujaldon, M., Nisbet, A., Amos, M., 2011. Parallelization strategies for ant colony optimisation on GPUs, in: Proceedings of the 25th International Conference on Parallel & Distributed Processing Symposium, pp. 339–346.

Chakroun, I., Melab, N., 2015. Towards a heterogeneous and adaptive parallel branch-and-bound algorithm. Journal of Computer and System Sciences 81, 72–84.

Chakroun, I., Melab, N., Mezmaz, M., Tuyttens, D., 2013a. Combining multi-core and GPU computing for solving combinatorial optimization problems. Journal of Parallel and Distributed Computing 73, 1563–1577.

Chakroun, I., Mezmaz, M., Melab, N., Bendjoudi, A., 2013b. Reducing thread divergence in a GPU-accelerated branch-and-bound algorithm. Concurrency and Computation-practice & Experience 25, 1121–1136.

Chaves-Gonzalez, J.M., Vega-Rodriguez, M.A., Gomez-Pulido, J.A., Sanchez-Perez, J.M., 2011. Optimizing a realistic large-scale frequency assignment problem using a new parallel evolutionary approach. Engineering Optimization 43, 813–842.

Christou, I.T., Vassilaras, S., 2013. A parallel hybrid greedy branch and bound scheme for the maximum distance-2 matching problem. Computers & Operations Research 40, 2387–2397.

Coelho, I.M., Munhoz, P.L.A., Ochi, L.S., Souza, M.J.F., Bentes, C., Farias, R.,

2016. An integrated CPU-GPU heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups. International Journal of Production Research 54, 945–962.

Cordeau, J.F., Maischberger, M., 2012. A parallel iterated tabu search heuristic for vehicle routing problems. Computers & Operations Research 39, 2033–2050.

Cotta, C., Talbi, E.G., Alba, E., 2005. Parallel hybrid metaheuristics. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 15. pp. 347–370.

Crainic, T., Toulouse, M., 2008. Learning and Intelligent Optimization. Springer, Berlin. volume 5315 of *Lecture Notes in Computer Science*. chapter Explicit and emergent cooperation schemes for search algorithms. pp. 95–109.

Crainic, T.G., 2008. Parallel solution methods for vehicle routing problems, in: The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, Boston, MA, pp. 171–198.

Crainic, T.G., 2018. Parallel Meta-heuristic Search, in: Martí, R., Panos, P., Resende, M.G. (Eds.), Handbook of Heuristics. Springer, Cham, pp. 1–39.

Crainic, T.G., 2019. Parallel metaheuristics and cooperative search, in: Gendreau, M., Potvin, J.Y. (Eds.), Handbook of Metaheuristics. Springer, Third edition. pp. 419–451.

Crainic, T.G., Davidović, T., Ramljak, D., 2014. Designing parallel metaheuristic methods, in: Despotovic-Zrakic, M., Milutinovic, V., Belic, A. (Eds.), Handbook of Research on High Performance and Cloud Computing in Scientific Research and Education. IGI Global, Hershey, PA. chapter 11, pp. 260–280.

Crainic, T.G., Gendreau, M., Potvin, J.Y., 2005. Parallel tabu search. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 13. pp. 289–313.

Crainic, T.G., Hail, N., 2005. Parallel metaheuristics applications. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 1. pp. 447–494.

Crainic, T.G., Le Cun, B., Roucairol, C., 2006. Parallel combinatorial optimization. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter Parallel Branch-and-Bound Algorithms. pp. 1–28.

Crainic, T.G., Toulouse, M., 2003. Parallel strategies for meta-heuristics, in: Handbook of metaheuristics. Springer, Boston, MA, pp. 475–513.

Crainic, T.G., Toulouse, M., 2010. Parallel meta-heuristics, in: Gendreau, M., Potvin, J.Y. (Eds.), Handbook of metaheuristics. Springer, Second edition. pp. 497–541.

Cung, V.D., Martins, S.L., Ribeiro, C.C., Roucairol, C., 2002. Strategies for the parallel implementation of metaheuristics, in: Essays and Surveys in Metaheuristics. Springer, Boston, MA, pp. 263–308.

Czapinski, M., 2010. Parallel simulated annealing with genetic enhancement for flowshop problem with CSUM. Computers & Industrial Engineering 59, 778–785.

Czapiński, M., 2013. An effective parallel multistart tabu search for quadratic assignment problem on CUDA platform. Journal of Parallel and Distributed Computing 73, 1461–1468.

Czapinski, M., Barnes, S., 2011. Tabu search with two approaches to parallel flowshop evaluation on CUDA platform. Journal of Parallel and Distributed Computing 71, 802–811.

Dai, C., Toulouse, M., Li, B.P.C., 2009. A multilevel cooperative tabu search algorithm for the covering design problem. The Journal of Combinatorial Mathematics and Combinatorial Computing , 35–65.

Davidović, T., Crainic, T.G., 2012. MPI parallelization of variable neighborhood search. Electronic Notes in Discrete Mathematics 39, 241–248.

Davidovic, T., Ramljak, D., Šelmic, M., Teodorovic, D., 2011. Mpi parallelization of bee colony optimization, in: Proc. 1st International Symposium & 10th Balkan Conference on Operational Research, pp. 193–200.

Deep, K., Sharma, S., Pant, M., 2010. Modified parallel particle swarm optimization for global optimization using message passing interface, in: Proceedings of the 5th International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), pp. 1451–1458.

Defersha, F.M., 2015. A simulated annealing with multiple-search paths and parallel computation for a comprehensive flowshop scheduling problem. International Transactions in Operational Research 22, 669–691.

Defersha, F.M., Chen, M., 2008. A parallel genetic algorithm for dynamic cell formation in cellular manufacturing systems. International Journal of Production Research 46, 6389–6413.

Defersha, F.M., Chen, M., 2010. A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups. International Journal of Advanced Manufacturing Technology 49, 263–279.

Defersha, F.M., Chen, M., 2012. Mathematical model and parallel genetic algorithm for hybrid flexible flowshop lot streaming problem. International Journal of Advanced Manufacturing Technology 62, 249–265.

Delevacq, A., Delisle, P., Gravel, M., Krajecki, M., 2013. Parallel ant colony optimization on graphics processing units. Journal of Parallel and Distributed Computing 73, 52–61.

Derbel, B., Humeauc, J., Liefooghe, A., Verel, S., 2014. Distributed localized bi-objective search. European Journal of Operational Research 239, 731–743.

Dias, B.H., Tomim, M.A., Marques Marcato, A.L., Ramos, T.P., Brandi, R.B.S., Da Silva Junior, I.C., Passos Filho, J.A., 2013. Parallel computing applied to the stochastic dynamic programming for long term operation planning of hydrothermal power systems. European Journal of Operational Research 229, 212–222.

Diaz, J., Munoz-Caro, C., Nino, A., 2012. A survey of parallel programming models and tools in the multi and many-core era. IEEE Transactions on parallel and distributed systems 23, 1369–1386.

Diego, F.J., Gómez, E.M., Ortega-Mier, M., García-Sánchez, Á., 2012. Parallel

CUDA architecture for solving de VRP with ACO, in: Industrial Engineering: Innovative Networks. Springer, Boston, MA, pp. 385–393.

Ding, K., Zheng, S., Tan, Y., 2013. A GPU-based parallel fireworks algorithm for optimization, in: Proceedings of the 15th annual conference on Genetic and evolutionary computation, pp. 9–16.

Djerrah, A., Le Cun, B., Cung, V.D., Roucairol, C., 2006. Bob++: Framework for solving optimization problems with branch-and-bound methods, in: Proceedings of the 15th IEEE International Conference on High Performance Distributed Computing, pp. 369–370.

Dobrian, F., Gebremedhin, A., Halappanavar, M., Pothen, A., et al., 2011. Distributed-memory parallel algorithms for matching and coloring, in: Proceedings of the 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, pp. 1971–1980.

Dongdong, G., Guanghong, G., Liang, H., Ni, L., 2010. Application of multicore parallel ant colony optimization in target assignment problem, in: Proceedings of the 2010 International Conference on Computer Application and System Modeling (ICCASM), pp. V3–514.

Dorigo, M., Stützle, T., 2004. Ant colony optimization. MIT Press.

Dorronsoro, B., Danoy, G., Nebro, A.J., Bouvry, P., 2013. Achieving superlinear performance in parallel multi-objective evolutionary algorithms by means of cooperative coevolution. Computers & Operations Research 40, 1552–1563.

Eckstein, J., Hart, W.E., Phillips, C.A., 2015. PEBBL: an object-oriented framework for scalable parallel branch and bound. Mathematical Programming Computation 7, 429–469.

Eskandarpour, M., Zegordi, S.H., Nikbakhsh, E., 2013. A parallel variable neighborhood search for the multi-objective sustainable post-sales network design problem. International Journal of Production Economics 145, 117–131.

Fabris, F., Krohling, R.A., 2012. A co-evolutionary differential evolution algorithm for solving min-max optimization problems implemented on GPU using c-CUDA. Expert Systems with Applications 39, 10324–10333.

Ferreiro, A.M., Garcia, J.A., Lopez-Salas, J.G., Vazquez, C., 2013. An efficient implementation of parallel simulated annealing algorithm in GPUs. Journal of Global Optimization 57, 863–890.

Figueira, J.R., Liefooghe, A., Talbi, E.G., Wierzbicki, A.P., 2010. A parallel multiple reference point approach for multi-objective optimization. European Journal of Operational Research 205, 390–400.

Fujimoto, N., Tsutsui, S., 2011. A highly-parallel TSP solver for a GPU computing platform, in: Proceedings of the 7th. International Conference on Numerical Methods and Applications, Springer-Verlag Berlin. pp. 264–271.

Galea, F., Le Cun, B., 2011. A parallel exact solver for the three-index quadratic assignment problem, in: Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, pp.

1940–1949.

Gao, J., He, G., Wang, Y., 2009. A new parallel genetic algorithm for solving multiobjective scheduling problems subjected to special process constraint. International Journal of Advanced Manufacturing Technology 43, 151–160.

Gendreau, M., Potvin, J.Y., et al. (Eds.), 2010. Handbook of metaheuristics. volume 146 of *International Series in Operations Research & Management Science*. Springer. third edition.

Gendreau, M., Potvin, J.Y., et al. (Eds.), 2019. Handbook of metaheuristics. volume 146 of *International Series in Operations Research & Management Science*. Springer.

Gendron, B., Crainic, T.G., 1994. Parallel branch-and-branch algorithms: survey and synthesis. Operations Research 42, 1042–1066.

Gerasch, T.E., Wang, P.Y., 1994. A survey of parallel algorithms for one-dimensional integer knapsack problems. INFOR: Information Systems and Operational Research 32, 163–186.

Gmys, J., Mezmaz, M., Melab, N., Tuyttens, D., 2016. A GPU-based branch-and-bound algorithm using integer-vector-matrix data structure. Parallel Computing 59, 119–139.

Gmys, J., Mezmaz, M., Melab, N., Tuyttens, D., 2017. Ivm-based parallel branch-and-bound using hierarchical work stealing on multi-GPU systems. Concurrency and Computation-practice & Experience 29.

Gomes, F.C., Meneses, C.N., Pardalos, P.M., Viana, G.V.R., 2008. A parallel multistart algorithm for the closest string problem. Computers & Operations Research 35, 3636–3643.

Groer, C., Golden, B., Wasil, E., 2011. A parallel algorithm for the vehicle routing problem. INFORMS Journal on Computing 23, 315–330.

Hadian, A., Shahrivari, S., Minaei-Bidgoli, B., 2012. Fine-grained parallel ant colony system for shared-memory architectures. International Journal of Computer Applications 53.

He, J., Chang, D., Mi, W., Yan, W., 2010. A hybrid parallel genetic algorithm for yard crane scheduling. Transportation Research Part E-logistics and Transportation Review 46, 136–155.

Hemmelmayr, V.C., 2015. Sequential and parallel large neighborhood search algorithms for the periodic location routing problem. European Journal of Operational Research 243, 52–60.

Herrera, J.F., Casado, L.G., Hendrix, E.M., Paulavicius, R., Ilinskas, J., 2013. Dynamic and hierarchical load-balancing techniques applied to parallel branch-and-bound methods, in: Proceedings of the 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 497–502.

Herrera, J.F.R., Salmeron, J.M.G., Hendrix, E.M.T., Asenjo, R., Casado, L.G., 2017. On parallel branch and bound frameworks for global optimization. Journal of Global Optimization 69, 547–560.

Hifi, M., Negre, S., Saadi, T., Saleh, S., Wu, L., 2014. A parallel large neighborhood search-based heuristic for the disjunctively constrained knapsack problem, in: Proceedings of the 2014 IEEE International Parallel & Distributed Processing Symposium Workshops, pp. 1547–1551.

Homberger, J., 2008. A parallel genetic algorithm for the multilevel unconstrained lot-sizing problem. INFORMS Journal on Computing 20, 124–132.

Hong, L., Zhong-hua, L., Xue-bin, C., 2010. Parallel computing for dynamic asset allocation based on the stochastic programming, in: Proceedings of the 2010 WASE International Conference on Information Engineering (ICIE), pp. 172–176.

Hou, N., He, F., Zhou, Y., Ai, H., 2017. A GPU-based tabu search for very large hardware/software partitioning with limited resource usage. Journal of Advanced Mechanical Design Systems and Manufacturing 11.

Huang, C.S., Huang, Y.C., Lai, P.J., 2012. Modified genetic algorithms for solving fuzzy flow shop scheduling problems and their implementation with CUDA. Expert Systems with Applications 39, 4999–5005.

Huebner, J., Schmidt, M., Steinbach, M.C., 2017. A distributed interior-point KKT solver for multistage stochastic optimization. INFORMS Journal on Computing 29, 612–630.

Hung, Y., Wang, W., 2012. Accelerating parallel particle swarm optimization via GPU. Optimization Methods & Software 27, 33–51.

Ibri, S., Drias, H., Nourelfath, M., 2010. A parallel hybrid ant-tabu algorithm for integrated emergency vehicle dispatching and covering problem. International Journal of Innovative Computing and Applications 2, 226–236.

INRIA, n.d. Paradiseo - A Software Framewok for Metaheuristics. http://paradiseo.gforge.inria.fr.

Ismail, M.A., Mirza, S.H., Altaf, T., 2011. A parallel and concurrent implementation of Lin-Kernighan heuristic (LKH-2) for solving traveling salesman problem for multi-core processors using SPC 3 programming model. International Journal of Advanced Computer Science and Applications .

Ismail, M.M., Abd El-Raoof, O., Abd El-Wahed, W.F., 2014. A parallel branch and bound algorithm for solving large scale integer programming problems. Applied Mathematics & Information Sciences 8, 1691–1698.

Izzo, D., Rucinski, M., Ampatzis, C., 2009. Parallel global optimisation metaheuristics using an asynchronous island-model, in: 2009 IEEE Congress on Evolutionary Computation, IEEE. pp. 2301–2308.

James, T., Rego, C., Glover, F., 2009. A cooperative parallel tabu search algorithm for the quadratic assignment problem. European Journal of Operational Research 195, 810–826.

Janiak, A., Janiak, W., Lichtenstein, M., 2008. Tabu search on GPU. Journal of Universal Computer Science 14, 2416–2427.

Janson, S., Merkle, D., Middendorf, M., 2005. Parallel ant colony algorithms. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 3. pp. 171–201.

Jin, J., Crainic, T.G., Løkketangen, A., 2011. A guided cooperative parallel tabu search for the capacitated vehicle routing problem, in: Proceedings of NIK 2011, pp. 49–60.

Jin, J., Crainic, T.G., Lokketangen, A., 2012. A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. European Journal of Operational Research 222, 441–451.

Jin, J., Crainic, T.G., Lokketangen, A., 2014. A cooperative parallel metaheuristic for the capacitated vehicle routing problem. Computers & Operations Research 44, 33–41.

Juan, A.A., Faulin, J., Jorba, J., Caceres, J., Manuel Marques, J., 2013. Using parallel & distributed computing for real-time solving of vehicle routing problems with stochastic demands. Annals of Operations Research 207, 43–65.

Kang, S., Kim, S.S., Won, J., Kang, Y.M., 2016. GPU-based parallel genetic approach to large-scale travelling salesman problem. Journal of Supercomputing 72, 4399–4414.

Kerkhove, L.P., Vanhoucke, M., 2017. A parallel multi-objective scatter search for optimising incentive contract design in projects. European Journal of Operational Research 261, 1066–1084.

Knysh, D.S., Kureichik, V.M., 2010. Parallel genetic algorithms: a survey and problem state of the art. Journal of Computer and Systems Sciences International 49, 579–589.

Koc, U., Mehrotra, S., 2017. Generation of feasible integer solutions on a massively parallel computer using the feasibility pump. Operations Research Letters 45, 652–658.

Kollias, G., Sathe, M., Schenk, O., Grama, A., 2014. Fast parallel algorithms for graph similarity and matching. Journal of Parallel and Distributed Computing 74, 2400–2410.

Ku, M.Y., Hu, M.H., Wang, M.J., 2011. Simulated annealing based parallel genetic algorithm for facility layout problem. International Journal of Production Research 49, 1801–1812.

Kumar, S., Misra, A., Tomar, R.S., 2011. A modified parallel approach to single source shortest path problem for massively dense graphs using CUDA, in: Proceedings of the 2nd International Conference on Computer and Communication Technology (ICCCT), pp. 635–639.

Laguna-Sanchez, G.A., Olguin-Carbajal, M., Cruz-Cortes, N., Barron-Fernandez, R., Alvarez-Cedillo, J.A., 2009. Comparative study of parallel variants for a particle swarm optimization algorithm implemented on a multithreading GPU. Journal of Applied Research and Technology 7, 292–309.

Lahrichi, N., Crainic, T.G., Gendreau, M., Rei, W., Crişan, G.C., Vidal, T., 2015. An integrative cooperative search framework for multi-decision-attribute combinatorial optimization: Application to the mdpvrp. European Journal of Operational Research 246, 400–412.

Lancinskas, A., Martinez Ortigosa, P., Zilinskas, J., 2015. Parallel optimization algorithm for competitive facility location. Mathematical Modelling and

Analysis 20, 619–640.

Lančinskas, A., Żilinskas, J., 2012. Solution of multi-objective competitive facility location problems using parallel NSGA-II on large scale computing systems, in: Proceedings of the 11th International Conference on Applied Parallel Computing, Springer-Verlag Berlin. pp. 422–433.

Lančinskas, A., Žilinskas, J., 2013. Parallel multi-objective memetic algorithm for competitive facility location, in: Proceedings of the 10th International Conference on Parallel Processing and Applied Mathematics, Springer-Verlag Berlin. pp. 354–363.

Lazarova, M., Borovska, P., 2008. Comparison of parallel metaheuristics for solving the TSP, in: Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing, p. 17.

Lei, D., Guo, X., 2015. A parallel neighborhood search for order acceptance and scheduling in flow shop environment. International Journal of Production Economics 165, 12–18.

Li, C.C., Lin, C.H., Liu, J.C., 2017. Parallel genetic algorithms on the graphics processing units using island model and simulated annealing. Advances in Mechanical Engineering 9.

Li, K., Liu, J., Wan, L., Yin, S., Li, K., 2015. A cost-optimal parallel algorithm for the 0-1 knapsack problem and its performance on multicore CPU and GPU implementations. Parallel Computing 43, 27–42.

Limmer, S., Fey, D., 2017. Comparison of common parallel architectures for the execution of the island model and the global parallelization of evolutionary algorithms. Concurrency and Computation: Practice and Experience 29.

Ling, C., Hai-Ying, S., Shu, W., 2012. A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem. Information Sciences 199, 31–42.

Liu, K.H., Kao, J.J., 2013. Parallelised branch-and-bound algorithm for raster-based landfill siting. Civil Engineering and Environmental Systems 30, 15–25.

Liu, Y.Y., Cho, W.K.T., Wang, S., 2016. Pear: a massively parallel evolutionary computation approach for political redistricting optimization and analysis. Swarm and Evolutionary Computation 30, 78–92.

Liu, Y.Y., Wang, S., 2015. A scalable parallel genetic algorithm for the generalized assignment problem. Parallel Computing 46, 98–119.

Lootsma, F.A., Ragsdell, K.M., 1988. State-of-the-art in parallel nonlinear optimization. Parallel Computing 6, 133–155.

Lou, Z., Reinitz, J., 2016. Parallel simulated annealing using an adaptive resampling interval. Parallel Computing 53, 23–31.

López, F.G., Torres, M.G., Batista, B.M., Perez, J.A.M., Vega, J.M.M., 2005. Parallel scatter search. John Wiley & Sons, Inc., Hoboken, New Jersey.. chapter 10. pp. 223–246.

Lu, H., Liu, J., Niu, R., Zhu, Z., 2014. Fitness distance analysis for parallel genetic algorithm in the test task scheduling problem. Soft Computing 18, 2385–2396.

Lubin, M., Martin, K., Petra, C.G., Sandikci, B., 2013. On parallelizing dual decomposition in stochastic integer programming. Operations Research Letters 41, 252–258.

Lubin, M., Petra, C.G., Anitescu, M., 2012. The parallel solution of dense saddle-point linear systems arising in stochastic programming. Optimization Methods and Software 27, 845–864.

Lucka, M., Melichercik, I., Halada, L., 2008. Application of multistage stochastic programs solved in parallel in portfolio management. Parallel Computing 34, 469–485.

Luna, F., Alba, E., Nebro, A.J., 2005. Parallel heterogeneous metaheuristics. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 17. pp. 395–422.

Luo, G.H., Huang, S.K., Chang, Y.S., Yuan, S.M., 2014. A parallel bees algorithm implementation on GPU. Journal of Systems Architecture 60, 271–279.

Luo, J., Hong, L.J., Nelson, B.L., Wu, Y., 2015. Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments. Operations Research 63, 1177–1194.

Luque, G., Alba, E., Dorronsoro, B., 2005. Parallel genetic algorithms. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 5. pp. 105–125.

Maischberger, M., Cordeau, J.F., 2011. Solving variants of the vehicle routing problem with a simple parallel iterated tabu search, in: Network optimization. Springer-Verlag Berlin, pp. 395–400.

Maleki, S., Musuvathi, M., Mytkowicz, T., 2016. Efficient parallelization using rank convergence in dynamic programming algorithms. Communications of the ACM 59, 85–92.

Martins, S.L., Ribeiro, C.C., 2006. Metaheuristics and applications to optimization problems in telecommunications, in: Handbook of Optimization in Telecommunications. Springer, Boston, MA, pp. 103–128.

Massobrio, R., Toutouh, J., Nesmachnow, S., Alba, E., 2017. Infrastructure deployment in vehicular communication networks using a parallel multiobjective evolutionary algorithm. International Journal of Intelligent Systems 32, 801–829.

McCreesh, C., Prosser, P., 2015. A parallel branch and bound algorithm for the maximum labelled clique problem. Optimization Letters 9, 949–960.

Melab, N., Luong, T., Boufaras, K., Talbi, E.G., 2011. Towards paradiseo-mo-gpu: a framework for gpu-based local search metaheuristics, in: International Work-Conference on Artificial Neural Networks, Springer. pp. 401–408.

Melab, N., Talbi, E.g., Cahon, S., Alba, E., Luque, G., 2006. Parallel metaheuristics: algorithms and frameworks. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 6. pp. 149–161.

Menendez, B., Pardo, E.G., Sanchez-Oro, J., Duarte, A., 2017. Parallel variable neighborhood search for the min-max order batching problem. International

Transactions in Operational Research 24, 635–662.

Mezmaz, M., Leroy, R., Melab, N., Tuyttens, D., 2014. A multi-core parallel branch-and-bound algorithm using factorial number system, in: Proceedings of the 28th International Symposium on Parallel and Distributed Processing, pp. 1203–1212.

Mezmaz, M., Melab, N., Kessaci, Y., Lee, Y.C., Talbi, E.G., Zomaya, A.Y., Tuyttens, D., 2011. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. Journal of Parallel and Distributed Computing 71, 1497–1508.

Mu, D., Wang, C., Zhao, F., Sutherland, J.W., 2016. Solving vehicle routing problem with simultaneous pickup and delivery using parallel simulated annealing algorithm. International Journal of Shipping and Transport Logistics 8, 81–106.

Munawar, A., Wahib, M., Munetomo, M., Akama, K., 2009. Hybrid of genetic algorithm and local search to solve MAX-SAT problem using nvidia CUDA framework. Genetic Programming and Evolvable Machines 10, 391–415.

Mussi, L., Daolio, F., Cagnoni, S., 2011. Evaluation of parallel particle swarm optimization algorithms within the CUDA architecture. Information Sciences 181, 4642–4657.

Nebro, A.J., Durillo, J.J., 2010. A study of the parallelization of the multi-objective metaheuristic MOEA/D, in: Proceedings of the 11th International Conference on Learning and Intelligent Optimization, Springer-Verlag Berlin. pp. 303–317.

Nebro, A.J., Luna, F., Talbi, E.g., Alba, E., 2005. Parallel multiobjective optimization. John Wiley & Sons, Inc., Hoboken, New Jersey.. chapter 16. pp. 371–394.

Nesmachnow, S., Cancela, H., Alba, E., 2012. A parallel micro evolutionary algorithm for heterogeneous computing and grid scheduling. Applied Soft Computing 12, 626–639.

Nesmachnow, S., Cancela, H., Alba, E., Chicano, F., 2005. Parallel metaheuristics in telecommunications. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 20. pp. 495–515.

Nowotniak, R., Kucharski, J., 2011. GPU-based massively parallel implementation of metaheuristic algorithms. Automatyka/Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie 15, 595–611.

Nwana, V., Mitra, 2000. Parallel mixed integer programming: a status review. Technical Report. Department of Mathematical Sciences, Brunel University.

Olensek, J., Tuma, T., Puhan, J., Burmen, A., 2011. A new asynchronous parallel global optimization method based on simulated annealing and differential evolution. Applied Soft Computing 11, 1481–1489.

Ozden, S.G., Smith, A.E., Gue, K.R., 2017. Solving large batches of traveling salesman problems with parallel and distributed computing. Computers & Operations Research 85, 87–96.

Pages-Bernaus, A., Perez-Valdes, G., Tomasgard, A., 2015. A parallelised distributed implementation of a branch and fix coordination algorithm. European Journal of Operational Research 244, 77–85.

Pardalos, P.M., Pitsoulis, L., Mavridou, T., Resende, M.G.C., 1995. Parallel search for combinatorial optimization: genetic algorithms, simulated annealing, tabu search and GRASP, in: Proceedings of the 2nd International Workshop on Parallel Algorithms for Irregularly Structured Problems, pp. 317–331.

Patvardhan, C., Bansal, S., Srivastav, A., 2016. Parallel improved quantum inspired evolutionary algorithm to solve large size quadratic knapsack problems. Swarm and Evolutionary Computation 26, 175–190.

Paulavičius, R., Žilinskas, J., 2009. Parallel branch and bound algorithm with combination of lipschitz bounds over multidimensional simplices for multicore computers, in: Parallel Scientific Computing and Optimization. Springer, Boston, MA, pp. 93–102.

Paulavicius, R., Zilinskas, J., Grothey, A., 2011. Parallel branch and bound for global optimization with combination of lipschitz bounds. Optimization Methods & Software 26, 487–498.

Pedemonte, M., Nesmachnow, S., Cancela, H., 2011. A survey on parallel ant colony optimization. Applied Soft Computing 11, 5181–5197.

Pedroso, D.M., Bonyadi, M.R., Gallagher, M., 2017. Parallel evolutionary algorithm for single and multi-objective optimisation: differential evolution and constraints handling. Applied Soft Computing 61, 995–1012.

Polacek, M., Benkner, S., Doerner, K.F., Hartl, R.F., 2008. A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. Business Research 1, 207–218.

Polat, O., 2017. A parallel variable neighborhood search for the vehicle routing problem with divisible deliveries and pickups. Computers & Operations Research 85, 71–86.

Ponz-Tienda, J.L., Salcedo-Bernal, A., Pellicer, E., 2017. A parallel branch and bound algorithm for the resource leveling problem with minimal lags. Computer-aided Civil and Infrastructure Engineering 32, 474–498.

Posypkin, M.A., Sigal, I.K., 2008. A combined parallel algorithm for solving the knapsack problem. Journal of Computer and Systems Sciences International 47, 543–551.

Pérez, J.A.M., Hansen, P., Mladenović, N., 2005. Parallel variable neighborhood search. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 11. pp. 247–266.

Qu, J., Liu, X., Sun, M., Qi, F., 2017. GPU-based parallel particle swarm optimization methods for graph drawing. Discrete Dynamics in Nature and Society .

Quan, Z., Wu, L., 2017. Design and evaluation of a parallel neighbor algorithm for the disjunctively constrained knapsack problem. Concurrency and Computation-practice & Experience 29.

Randall, M., Lewis, A., 2002. A parallel implementation of ant colony optimization. Journal of Parallel and Distributed Computing 62, 1421–1432.

Rashid, H., Novoa, C., Qasem, A., 2010. An evaluation of parallel knapsack algorithms on multicore architectures., in: Proceedings of the 2010 International Conference on Scientific Computing, pp. 230–235.

Ravetti, M.G., Riveros, C., Mendes, A., Resende, M.G.C., Pardalos, P.M., 2012. Parallel hybrid heuristics for the permutation flow shop problem. Annals of Operations Research 199, 269–284.

Redondo, J.L., Fernandez, J., Garcia, I., Ortigosa, P.M., 2008. Parallel algorithms for continuous competitive location problems. Optimisation Methods & Software 23, 779–791.

Redondo, J.L., Garcia, I., Ortigosa, P.M., 2011. Parallel evolutionary algorithms based on shared memory programming approaches. Journal of Supercomputing 58, 270–279.

Redondo, J.L., Marin, A., Ortigosa, P.M., 2016. A parallelized lagrangean relaxation approach for the discrete ordered median problem. Annals of Operations Research 246, 253–272.

Resende, M.G.C., Ribeiro, C.C., 2005. Parallel greedy randomized adaptive search procedures. John Wiley & Sons, Inc., Hoboken, New Jersey.. chapter 14. pp. 315–346.

Roberge, V., Tarbouchi, M., Labonte, G., 2013. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. IEEE Transactions on Industrial Informatics 9, 132–141.

Rossbory, M., Reisner, W., 2013. Parallelization of algorithms for linear discrete optimization using paraphrase, in: Proceedings of the 24th International Workshop on Database and Expert Systems Applications (DEXA), pp. 241–245.

Rudek, R., 2014. Exact and parallel metaheuristic algorithms for the single processor total weighted completion time scheduling problem with the sum-of-processing-time based models. Computers & Operations Research 46, 91–101.

Rudolph, G., 2005. Parallel evolution strategies. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 7. pp. 155–169.

Sancı, S., İşler, V., 2011. A parallel algorithm for UAV flight route planning on GPU. International Journal of Parallel Programming 39, 809–837.

Sanjuan-Estrada, J., Casado, L.G., García, I., 2011. Adaptive parallel interval global optimization algorithms based on their performance for non-dedicated multicore architectures, in: Proceedings of the 19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 252–256.

Santos, L., Madeira, D., Clua, E., Martins, S., Plastino, A., 2010. A parallel GRASP resolution for a GPU architecture, in: Proceedings of the 7th International Conference on Metaheuristics and Nature Inspired Computing, p. META10.

Sathe, M., Schenk, O., Burkhart, H., 2012. An auction-based weighted matching implementation on massively parallel architectures. Parallel Computing 38, 595–614.

Scheerlinck, K., Vernieuwe, H., De Baets, B., 2012. Zadeh's extension principle for continuous functions of non-interactive variables: a parallel optimization approach. IEEE Transactions on Fuzzy Systems 20, 96–108.

Schulz, C., Hasle, G., Brodtkorb, A.R., Hagen, T.R., 2013. GPU computing in discrete optimization: Part ii: Survey focused on routing problems. EURO Journal on Transportation and Logistics 2, 159–186.

Shylo, O.V., Middelkoop, T., Pardalos, P.M., 2011. Restart strategies in optimization: parallel and serial cases. Parallel Computing 37, 60–68.

Silva, J.M.N., Boeres, C., Drummond, L.M.A., Pessoa, A.A., 2015. Memory aware load balance strategy on a parallel branch-and-bound application. Concurrency and Computation-practice & Experience 27, 1122–1144.

Skinderowicz, R., 2016. The GPU-based parallel ant colony system. Journal of Parallel and Distributed Computing 98, 48–60.

Stanojevic, P., Maric, M., Stanimirovic, Z., 2015. A hybridization of an evolutionary algorithm and a parallel branch and bound for solving the capacitated single allocation hub location problem. Applied Soft Computing 33, 24–36.

Stivala, A., Stuckey, P.J., Garcia De La Banda, M., Hermenegildo, M., Wirth, A., 2010. Lock-free parallel dynamic programming. Journal of Parallel and Distributed Computing 70, 839–848.

Subotic, M., Tuba, M., Stanarevic, N., 2011. Different approaches in parallelization of the artificial bee colony algorithm. International Journal of mathematical models and methods in applied sciences 5, 755–762.

Subramanian, A., Drummond, L.M.A., Bentes, C., Ochi, L.S., Farias, R., 2010. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Computers & Operations Research 37, 1899–1911.

Talbi, E.G. (Ed.), 2006. Parallel combinatorial optimization. John Wiley & Sons.

Talbi, E.G., 2009. Metaheuristics: from design to implementation. John Wiley & Sons.

Tan, G., Sun, N., Gao, G.R., 2009. Improving performance of dynamic programming via parallelism and locality on multicore architectures. IEEE Transactions on Parallel and Distributed Systems 20, 261–274.

Tan, Y., Ding, K., 2016. A survey on GPU-based implementation of swarm intelligence algorithms. IEEE Transactions on Cybernetics 46, 2028–2041.

Taoka, S., Takafuji, D., Watanabe, T., 2008. Enhancing PC cluster-based parallel branch-and-bound algorithms for the graph coloring problem. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E91A, 1140–1149.

Thiruvady, D., Ernst, A.T., Singh, G., 2016. Parallel ant colony optimization for resource constrained job scheduling. Annals of Operations Research 242, 355–372.

Tosun, U., Dokeroglu, T., Cosar, A., 2013. A robust island parallel genetic algorithm for the quadratic assignment problem. International Journal of Production Research 51, 4117–4133.

Toulouse, M., Crainic, T.G., Sansó, B., 2004. Systemic behavior of cooperative search algorithms. Parallel Computing 30, 57–79.

Toulouse, M., Crainic, T.G., Thulasiraman, K., 2000. Global optimization properties of parallel cooperative search algorithms: a simulation study. Parallel Computing 26, 91–112.

Tran, Q.N., 2010. Designing efficient many-core parallel algorithms for all-pairs shortest-paths using CUDA, in: Proceedings of the 7th International Conference on Information Technology: New Generations (ITNG), pp. 7–12.

Trelles, O., Rodriguez, A., 2005. Bioinformatics and parallel metaheuristics. John Wiley & Sons, Inc., Hoboken, New Jersey. chapter 21. pp. 517–549.

Tsutsui, S., 2008. Parallel ant colony optimization for the quadratic assignment problems with symmetric multi processing, in: Proceedings of the 6th International Conference Ant Colony Optimization and Swarm Intelligence, Springer-Verlag Berlin. pp. 363–370.

Tu, W., Li, Q., Li, Q., Zhu, J., Zhou, B., Chen, B., 2017. A spatial parallel heuristic approach for solving very large-scale vehicle routing problems. Transactions in Gis 21, 1130–1147.

Umbarkar, A., Joshi, M.S., Hong, W.C., 2014. Multithreaded parallel dual population genetic algorithm (mpdpga) for unconstrained function optimizations on multi-core system. Applied Mathematics and Computation 243, 936–949.

Vallada, E., Ruiz, R., 2009. Cooperative metaheuristics for the permutation flowshop scheduling problem. European Journal of Operational Research 193, 365 – 376.

Van Luong, T., Melab, N., Talbi, E.G., 2013. GPU computing for parallel local search metaheuristic algorithms. IEEE transactions on computers 62, 173–185.

Van Luong, T., Taillard, E., Melab, N., Talbi, E.G., 2012. Parallelization strategies for hybrid metaheuristics using a single GPU and multi-core resources, in: Proceedings of the 12th International Conference on Parallel Problem Solving from Nature, Springer-Verlag Berlin. pp. 368–377.

Vidal, P., Alba, E., Luna, F., 2017. Solving optimization problems using a hybrid systolic search on GPU plus CPU. Soft Computing 21, 3227–3245.

Vu, T.t., Derbel, B., 2016. Parallel branch-and-bound in multi-core multi-CPU multi-GPU heterogeneous environments. Future Generation Computer Systems-the International Journal of Escience 56, 95–109.

Wang, C., Mu, D., Zhao, F., Sutherland, J.W., 2015. A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup-delivery and time windows. Computers & Industrial Engineering 83, 111–122.

Wang, D., Wu, C.H., Ip, A., Wang, D., Yan, Y., 2008. Parallel multi-population particle swarm optimization algorithm for the uncapacitated facility location problem using OpenMP, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 1214–1218.

Wang, K., Shen, Z., et al., 2012. A GPU-based parallel genetic algorithm for generating daily activity plans. IEEE Trans. Intelligent Transportation Systems 13, 1474–1480.

Weber, M., Neri, F., Tirronen, V., 2011. Shuffle or update parallel differential evolution for large-scale optimization. Soft Computing 15, 2089–2107.

Wei, K.c., Sun, X., Chu, H., Wu, C.C., 2017. Reconstructing permutation table to improve the tabu search for the PFSP on GPU. Journal of Supercomputing 73, 4711–4738.

Xhafa, F., Duran, B., 2008. Parallel memetic algorithms for independent job scheduling in computational grids, in: Recent advances in evolutionary computation for combinatorial optimization. Springer-Verlag Berlin, pp. 219–239.

Xu, Y., Ralphs, T.K., Ladanyi, L., Saltzman, M.J., 2009. Computational experience with a software framework for parallel integer programming. INFORMS Journal on Computing 21, 383–397.

Yang, Q., Fang, L., Duan, X., 2016. RMACO: a randomly matched parallel ant colony optimization. World Wide Web: Internet and Web Information Systems 19, 1009–1022.

Yazdani, M., Amiri, M., Zandieh, M., 2010. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. Expert Systems with Applications 37, 678–687.

You, Y.S., 2009. Parallel ant system for traveling salesman problem on GPUs, in: Proceedings of 11th. International Conference on Genetic and Evolutionary Computation, pp. 1–2.

Yu, B., Yang, Z., Sun, X., Yao, B., Zeng, Q., Jeppesen, E., 2011a. Parallel genetic algorithm in bus route headway optimization. Applied Soft Computing 11, 5081–5091.

Yu, B., Yang, Z.Z., Xie, J.X., 2011b. A parallel improved ant colony optimization for multi-depot vehicle routing problem. Journal of the Operational Research Society 62, 183–188.

Yu, W.J., Li, J.Z., Chen, W.N., Zhang, J., 2017. A parallel double-level multi-objective evolutionary algorithm for robust optimization. Applied Soft Computing 59, 258–275.

Ze-Shu, R.A.O., Wan-Ying, Z.H.U., ZHANG, K., 2017. Solving graph coloring problem using parallel discrete particle swarm optimization on CUDA. DEStech Transactions on Engineering and Technology Research .

Zhang, X.Y., Zhang, J., Gong, Y.J., Zhan, Z.H., Chen, W.N., Li, Y., 2016. Kuhnmunkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks. IEEE Transactions on Evolutionary Computation 20, 695–710.

Zhang, Y., Wang, S., Ji, G., 2015. A comprehensive survey on particle swarm optimization algorithm and its applications. Mathematical Problems in Engineering 2015.

Zhao, J., Liu, Q., Wang, W., Wei, Z., Shi, P., 2011. A parallel immune algorithm for traveling salesman problem and its application on cold rolling scheduling. Information Sciences 181, 1212–1223.

Zhou, Y., He, F., Hou, N., Qiu, Y., 2017. Parallel ant colony optimization on multi-core SIMD CPUs. Future Generation Computer Systems .

Zhu, W., Curry, J., 2009. Parallel ant colony for nonlinear function optimization with graphics hardware acceleration, in: Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics, pp. 1803–1808.

Appendix A. Literature selection process

When invited by the editorial board of *European Journal of Operational Research* in 2018, we were recommended to concentrate on the last decade of literature whenever possible. Following this recommendation is particularly reasonable for the body of literature on parallel optimization in OR because it accounts for a massive growth in computing performance in this period and resulting substantial advances of studies published regarding algorithmic parallelization, parallel software implementation and achieved computational results.

We conducted a title search in the most renowned OR journals. More specifically, we considered those 49 OR journals which are ranked "A+", "A", "B" or "C" in the German VHB-JOURQUAL 3 ranking of the German Academic Association for Business Research (German Academic Association for Business Research (VHB)); a complete list of these journals is included in Table A.5. As we expected to find research related to parallel optimization in OR also in journals that are dedicated to parallel computing, we included the following four journals in our search: *Journal of Parallel and Distributed Computing*, *International Journal of Parallel Programming*, *Parallel Programming* and *Parallel Processing and Applied Mathematics*. We used *Web of Science* to conduct a title search for both sets of journals, using the following search string:

> (parallel* OR distributed OR "'shared memory" OR MPI OR OpenMP OR CUDA OR GPU OR SMP) AND NOT "parallel machine"

4OR	Journal of Decision Systems
Annals of Operations Research	Journal of Economic Dynamics & Control
Artificial Intelligence	Journal of Forecasting
Asia-Pacific Journal of Operational Research	Journal of Heuristics
Central European Journal of Operations Research	Journal of Operations Management
Computers and Operations Research	Journal of Revenue and Pricing Management
Computers in Industry	Journal of Risk and Uncertainty
Decision Sciences	Journal of Scheduling
Decision Support Systems	Journal of the Operational Research Society
Discrete Applied Mathematics	Logistics Research
EURO Journal on Transportation and Logistics	Management Information Systems Quarterly
European Journal of Operational Research	Managerial and Decision Economics
Flexible Services and Manufacturing Journal	Manufacturing & Service Operations Management
Group Decision and Negotiation	Mathematical Methods of Operations Research
IEEE Transactions on Systems, Man, and Cybernetics	Mathematical Programming
IIE Transactions	Mathematics of Operations Research
Information Systems Research	Naval Research Logistics
INFORMS Journal on Computing	Operations Research
Interfaces	Operations Research Letters
International Journal of Forecasting	OR Spectrum
International Journal of Information Technology & Decision Making	SIAM Journal on Computing
International Journal of Operations & Production Management	System Dynamics Review
International Journal of Operations Research	Transportation Research Part B: Methodological
International Journal of Production Economics	Transportation Science
International Journal of Production Research	

Table A.5: Operation research journals considered in literature selection process (in alphabetical order)

Acknowledging that research on parallel optimization relevant to the OR discipline is likely to be published also in journals of other disciplines and in conference proceedings and books, we also conducted a title search using *Web of Science Core Collection* without any restrictions regarding the publication outlet. However, we needed to adjust the search string in order keep the resulting list of articles manageable. The search strings that we used is as follows:

- "parallel* optimization" OR "parallel* branch" OR "parallel* discrete" OR "parallel heuristic" OR "parallel exact" OR "parallel meta" OR "parallel genetic" OR "parallel tabu" OR "parallel evolutionary" OR "parallel* ant colony" OR "parallel* simulated annealing" OR "parallel* variable neighborhood search" OR "parallel* Greedy Randomized Adaptive Search Procedures" OR "parallel* scatter search" OR "parallel* dynamic programming"
- (MPI OR OpenMP OR CUDA OR GPU) AND (heuristic* OR exact OR meta OR genetic OR branch OR optimization OR discrete OR tabu)
- (parallel* AND algorithm) AND (knapsack OR transport OR logistics OR evolutionary)

We also conducted a backward search of reference sections of literature reviews we identified (see the introduction of this article).

Overall, our literature search returned more than 1,100 entries. With the support of a PhD and several student workers, we used the title of an article to decide whether it should be excluded from further analysis due to a missing fit with the scope of this review, resulting in a preliminary list of 238 entries . Finally, with the help of the student workers we analyzed the content of each of these articles and excluded further 83 entries for a variety of reasons, including a missing fit with scope and the use of languages other than English. Finally, we conducted a backward search of reference sections of the remaining 155 articles to mitigate the risk of overlooking relevant studies: in a first step, we selected potentially relevant articles based on their title; in a second step, we analyzed the selected articles by inspecting the full text to decide whether they should be included in the final set of considered articles or not; this procedure yielded 50 additional articles. Overall, the ultimate set of articles, referred to as *our sample*, consists of 206 computational studies on parallel optimization in OR published between 2008 and 2017.

Appendix B. Coding of computational parallelization studies

This section contains the detailed coding results of our sample with the exception of three studies: Östermark (2014, 2015) do not explicit the algorithm parallelized; Bozejko (2012) parallelizes the problem-specific evaluation of objective function but no overall algorithm is considered. To sum up, the tables in this section include 203 studies of the full sample (206 studies).

The articles are grouped along types of algorithms, with Table B.6 addressing exact methods, Table B.7 addressing single-solution based metaheuristics, Table B.8 addressing population-based metaheuristics, Table B.9 addressing hybrid metaheuristics, and Table B.10 addressing problem-specific heuristics, other heuristics, and matheuristics. Unsurprisingly, not all studies included in our sample provide sufficiently precise details that allow coding all attributes. In cases where incomplete or ambiguous information is provided , we use the value "n/a". We need to point to two exceptions from this rule: 1) in the column "Process and search control", which show a triple classification, the usage of "n/a" for one or more of the three classes may confuse the reader. Thus, we prefer to use the symbol "?" where information is not available or ambiguous, or where our classification is not applicable (e.g., in reference (Derbel et al., 2014), a semi-synchronous mode is used because MPI-synchronization occurs at a pairwise level but not at a global level (p. 15)). 2) The entry "n/a" in the "Scalability" column has a more sophisticated interpretation, which we unfold in the text below.

The entries in the columns labeled "Problem" and "Algorithm" use the abbreviations as shown in Table 2 in the main text of the article. Entries in columns labeled "Parallelization strategy", "Process & search control", "Communication topology" and "Programming model" are used as described in the main text.

The column "Scalability" covers both speedup and efficiency. It shows different types of entries: speedup that is qualified by its type of efficiency is provided in the form "sublinear (n=2-16)", for example, where the range of n indicating the numbers of parallel processing units used. Speedup that varies between (sub)linear and superlinear depending on tested instances is described accordingly. Speedup achieved with GPGPUs is given as a single value or as an interval. We do not qualify speedup in this case as the number of parallel working units (usually GPGPU threads) needs to be interpreted different from that counting other parallel working units (CPU threads, processes) because they differ substantially from a technological perspective. Also, for the same reason, the determination of efficiency of parallelization should not be computed as the ratio of speedup and the number of parallel processing units. The entry "n/a" in the "Scalability" column is an umbrella type and can have several different meanings described below. When more than one experiment has been conducted (e.g, applying different (versions of) algorithms, different (sets of) benchmark instances, and/or different programming models), speedup information is numbered.

Reasons for labeling scalability as "n/a" turned out to be appropriate for manifold reasons:

- Times are compared with theoretical serial times.
- Speedup is related to other parallel executed algorithms or to parallel execution of the same algorithm (for example, because the execution on a single processing unit was practically infeasible due to time limitations); i.e., we report only speedups (weak or relative) related to serial executions of algorithms.
- The type of reference execution is unknown.

- No speedup values are reported or tedious work is necessary to determine them from data reported.
- Speedup values are provided in in supplementary material which is inaccessible.
- Speedup values only refer to parts of algorithms.
- Running times must not be compared as i) different (hardware) machines/computing environments are used, or ii) different levels of objective functions are achieved by reference execution(s) and execution of parallel algorithm.
- Parallelization is conducted in a virtual environment where no physical parallelization occurs. Then, execution times are hardly comparable as parallel execution times will often be larger than sequential times due to parallelization overhead.

We do not qualify speedup (as "linear", for example) in the case of GPGPU as programming model as the number of parallel working units (usually GPGPU threads) needs to be interpreted different from that counting other parallel working units (CPU threads, processes) because they differ substantially from a technological perspective. Also, for the same reason, the determination of efficiency of parallelization should not be computed as the ratio of speedup and the number of parallel processing units.

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Mezmaz et al., 2014)	FSSP	B-a-X	domain decomposition	1. 1C/C/MPSS, 2. pC/C/MPSS	1.master-slave, 2. ring	n/a	n/a
(Chakroun et al., 2013b)	FSSP	B-a-X	low-level	1C/RS/SPSS	master-slave	GPGPU	[40.52-60.64]
(Herrera et al., 2017)	BFP	B-a-X	domain decomposition	pC/C/MPSS	n/a	threads	1. sublinear (n=1-16), 2. and 3. linear (n=1-16)
(Taoka et al., 2008)	GTP	B-a-X	domain decomposition	pC/C/MPSS	mesh	message passing	linear (n=4-16)
(Ponz-Tienda et al., 2017)	Other	B-a-X	domain decomposition	1C/C/MPSS	master-slave	hybrid (message passing + shared memory)	superlinear (n=160)
(Ismail et al., 2014)	KP	B-a-X	domain decomposition	pC/C/MPSS	tree	message passing	[1.9-7.3] (unreported n)
(Paulavicius et al., 2011)	Other	B-a-X	domain decomposition	1C/C/MPSS	n/a	1. shared memory, 2. message passing	sublinear (n=2-16)
(Christou and Vassilaras, 2013)	GTP	B-a-X	domain decomposition	1C/C/MPSS	n/a	threads	sublinear (n=2-16)
(McCreesh and Prosser, 2015)	GTP	B-a-X	domain decomposition	1C/C/MPSS	n/a	threads	linear (n=4)
(Eckstein et al., 2015)	Other	B-a-X	domain decomposition	1C/C/MPSS	master-slave	message passing	linear (n=2-8192)
(Carvajal et al., 2014)	MILP	B-a-X	1. indep. multi-search, 2./3. coop. multi-search	1. pC/RS/SPDS, 2. pC/KC/SPDS, 3. pC/?/SPDS	master-slave	message passing	n/a
(Borisenko et al., 2017)	Other	B-a-X	domain decomposition	1C/C/MPSS	master-slave	GPGPU	[1.67-5.79]
(Gmys et al., 2017)	FSSP, TSP, Other	B-a-X	domain decomposition	1C/C/MPSS	n/a	GPGPU	n/a
(Liu and Kao, 2013)	Other	B-a-X	domain decomposition	1C/C/MPSS	master-slave	shared memory (read/write lock used)	linear (n=5)
(Bak et al., 2011)	Other	B-a-X	domain decomposition	1C/C/MPSS	master-slave	message passing	varies between instances (n=2-20)
(Gmys et al., 2016)	FSSP	B-a-X	domain decomposition	1C/C/MPSS	n/a	GPGPU	n/a
(Silva et al., 2015)	Other	B-a-X	domain decomposition	pC/C/MPSS	tree	hybrid (shared memory + message passing)	varies between instances (n=16), sublinear/linear (n=32-64)
(Barreto and Bauer, 2010)	MILP	B-a-X	domain decomposition	1C/C/MPSS	master-slave	1. message passing, 2. shared-memory	sublinear (n=2-100)
(Vu and Derbel, 2016)	FSSP	B-a-X	hybrid (low-level + domain decomposition)	n/a	fully connected mesh	1. message passing, 2. hybrid (shared memory + message passing)	sublinear/linear (n=1-16 GPUs, m=1-512 CPUs)
(Chakroun and Melab, 2015)	FSSP	B-a-X	domain decomposition	pC/C/MPSS	n/a	1. n/a, 2. threads, 3. n/a, 4. n/a	1. [80-160], 2.linear (n=2-6 CPUs), 3. [79.42-124.1], 4.[198.55-222.02]
(Paulavičius and Žilinskas, 2009)	Other	B-a-X	low-level	1C/RS/SPSS	master-slave	shared memory	sublinear (n=2-4)
(Posypkin and Sigal, 2008)	KP	B-a-X	domain decomposition	pC/C/MPSS	tree	message passing	sublinear (n=8-32)
(Chakroun et al., 2013a)	FSSP	B-a-X	domain decomposition	pC/C/MPSS	n/a	GPGPU	[76.96-170.69]

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Aitzai and Boudhar, 2013)	JSSP	B-a-X	domain decomposition	1.pC/C/MPSS, 1C/C/MPSS	hybrid (master-slave + ring)	message passing	varies between instances (n=3-4)
(Ozden et al., 2017)	TSP	B-a-X	domain decomposition	1C/C/MPSS	master-slave	message passing	n/a
(Cauley et al., 2011)	Other	B-a-X	domain decomposition	pC/C/MPSS	hybrid (master-slave + ring)	message passing	n/a
(Xu et al., 2009)	MILP	B-a-X	domain decomposition	1C/C/MPSS	master-slave	message passing	n/a
(Aldasoro et al., 2017)	SOP	B-a-X	domain decomposition	pC/C/MPSS	tree	hybrid (message passing + threads)	n/a
(Pages-Bernaus et al., 2015)	MILP	B-a-X	domain decomposition	1C/C/MPSS	master-slave	message passing	sublinear (n=12)
(Lubin et al., 2013)	SOP	B-a-X	domain decomposition	1C/C/MPSS	n/a	message passing	sublinear (n=8-32)
(Huebner et al., 2017)	SOP	IPM	low-level	1C/RS/SPSS	fully connected mesh	message passing	sublinear (n=8-64)
(Dias et al., 2013)	Other	DP	low-level	1C/RS/SPSS	master-slave	message passing	sublinear (n=8-256)
(Aldasoro et al., 2015)	SOP	DP	1. low-level, 2. coop. multi-search	1. 1C/RS/SPSS, 2.pC/RS/SPDS	1. tree, 2. master-slave	message passing	sublinear (n=12)
(Maleki et al., 2016)	Other	DP	hybrid (low-level + domain decomposition)	pC/RS/MPDS	n/a	1. shared memory, 2. distributed memory	sublinear (n=(1 or 8)-128) (probably wrt. distributed memory)
(Tan et al., 2009)	n/a	DP	low-level	1C/RS/SPSS	tree	threads	sublinear (n=4-64)
(Stivala et al., 2010)	KP, GTP, Other	DP	coop. multi-search	pC/C/SPDS	n/a	shared memory	mostly sublinear (n<=32)
(Boyer et al., 2012)	KP	DP	low-level	1C/RS/SPSS	tree	GPGPU	[18.9-26.05]
(Boschetti et al., 2016)	Other	DP	low-level	1C/RS/SPSS	tree	hybrid (shared memory + GPGPU)	n/a
(Kollias et al., 2014)	GTP	PSEA	low-level	1C/RS/SPSS	master-slave	hybrid	linear (n<=1024)
(Bozdağ et al., 2008)	GTP	PSEA	domain decomposition	pC/RS/MPSS	mesh	message passing	sublinear (n<=40)
(Li et al., 2015)	KP	PSEA	low-level	n/a	n/a	1. shared memory, 2. GPGPU	1. sublinear or linear (varies between instances) (n=2-16), 2. [2-16]
(Adel et al., 2016)	JSSP	B-a-X	hybrid (low-level+domain decomposition)	n/a	master-slave	GPGPU	[0.31-65.5]
(Borisenko et al., 2011)	Other	B-a-X	domain decomposition	1C/C/MPSS	n/a	hybrid (shared memory + message passing)	superlinear (n<=32)
(Boukedjar et al., 2012)	KP	B-a-X	domain decomposition	1C/C/MPSS	n/a	GPGPU	[3.84-9.27]
(Carneiro et al., 2011)	TSP	B-a-X	domain decomposition	pC/C/MPSS	n/a	1. shared memory, 2. GPGPU	1. sublinear (n=4), 2. [7.61-10.69]
(Galea and Le Cun, 2011)	AP	B-a-X	1. low-level, 2. domain decomposition	1. 1C/RS/SPSS, 2. n/a	ring	1. threads, 2. SIMD	superlinear (n<=24)

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Herrera et al., 2013)	Other	B-a-X	domain decomposition	1. 1C/C/MPSS, 2. pC/C/MPSS	1. master-slave, 2. tree	hybrid (message passing + threads)	sublinear (n=32-128)
(Hong et al., 2010)	Other	IPM	low-level	1C/RS/SPSS	n/a	message passing	linear (n<=16)
(Kumar et al., 2011)	GTP	DP	1. n/a, 2. low-level	1. n/a, 2. 1C/RS/SPSS	n/a	1. threads, 2. GPGPU	1. sublinear (n=2), 2. [10.27-13.82]
(Lubin et al., 2012)	SOP	IPM	low-level	1C/RS/SPSS	n/a	message passing	sublinear (n=32-2048)
(Lucka et al., 2008)	Other	IPM	low-level	1C/RS/SPSS	master-slave	message passing	sublinear (n<=16)
(Rashid et al., 2010)	KP	DP	low-level	1C/RS/SPSS	1. master-slave, 2. mesh	shared memory	sublinear (n<=16)
(Rossbory and Reisner, 2013)	MILP	PSEA	domain decomposition	1C/RS/SPSS	master-slave	n/a	sublinear (n=12)
(Sanjuan-Estrada et al., 2011)	BFP, Other	B-a-X	domain decomposition	pC/C/MPSS	n/a	threads	linear (1<n<4), sublinear (4<n<128), n=average no. of running threads
(Tran, 2010)	GTP	DP	low-level	1C/RS/SPSS	n/a	GPGPU	two algorithm versions: 1. [48-52], 2. [900-2500]

Table B.6: Computational parallelization studies (exact algorithms)

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Thiruvady et al., 2016)	JSSP	SA	1a. low-level, 1b. coop. multi-search, 2. coop. multi-search	1a. 1C/RS/SPSS, 1b. pC/C/?PDS, 2. pC/C/SPDS	n/a	shared memory	n/a
(Rudek, 2014)	MSP	TS	low-level	1C/RS/SPSS	master-slave	threads	sublinear (n=8)
(Rudek, 2014)	MSP	SA	indep. multi-search	pC/RS/SPDS	master-slave	threads	sublinear (n=8)
(Yazdani et al., 2010)	JSSP	VNS	low-level	1C/RS/SPSS	master-slave	n/a	n/a
(Lei and Guo, 2015)	FSSP	VNS	coop. multi-search	pC/C/MPDS	bidirectional (two nodes)	message passing	sublinear (n=2)
(Davidović and Crainic, 2012)	MSP	VNS	1. low-level, 2. domain decomposition, 3. coop. multi-search, 4. coop. multi-search; 5. coop. multi-search	1. 1C/RS/SPSS, 2. 1C/C/MPSS, 3. pC/C/SPDS, 4. pC/C/SPDS, 5. pC/C/MPDS	1.- 2. n/a, 3.- 4. master-slave, 5.ring	message passing	n/a
(Quan and Wu, 2017)	KP	VNS	domain decomposition	1C/RS/MPDS	master-slave	message passing	n/a
(Menendez et al., 2017)	Other	VNS	low-level	1C/RS/SPSS	master-slave	threads	sublinear (n=8)
(Eskandarpour et al., 2013)	Other	VNS	low-level	1C/RS/SPSS	master-slave	threads	n/a
(Coelho et al., 2016)	VRP	VNS	low-level	1C/RS/SPSS	n/a	GPGPU	[0.93-14.49]
(Polat, 2017)	VRP	VNS	n/a	pC/C/MPSS	star	n/a	sublinear (n=6)
(Tu et al., 2017)	VRP	VNS	low-level	1C/RS/SPSS	n/a	shared memory	sublinear (n=2-32)
(Defersha, 2015)	FSSP	SA	domain decomposition	1C/KS/SPMS	master-slave	message passing	n/a
(Mu et al., 2016)	VRP	SA	coop. multi-search	pC/RS/SPDS	master-slave	threads	n/a
(Wang et al., 2015)	VRP	SA	coop. multi-search	pC/RS/SPDS	master-slave	threads	n/a
(Ferreiro et al., 2013)	Other	SA	1. indep. multi-search, 2. coop. multi-search	pC/RS/SPDS	1. unconnected, 2. master-slave	GPGPU	[73.44-269.46]
(Lou and Reinitz, 2016)	BFP	SA	coop. multi-search	pC/?/?PDS	n/a	message passing	sublinear (n=24-192)
(Banos et al., 2016)	VRP	SA	1. domain decomposition, 2. cooperative multi-search	1. 1C/KS/SPDS, 2. pC/C/MPDS	master-slave	1. shared memory, 2. message passing	n/a
(Jin et al., 2012)	VRP	TS	coop. multi-search	pC/RS/SPDS	master-slave	threads	n/a
(Bozejko et al., 2017)	JSSP	TS	low-level	1C/RS/SPSS	master-slave	shared memory	sublinear (n=2-224)
(Hou et al., 2017)	Other	TS	low-level	1C/RS/SPSS	n/a	GPGPU	[4]
(Bozejko et al., 2013)	FSSP	TS	low-level	1C/RS/SPSS	master-slave	SIMD	linear or superlinear (varies between instances) (n=2)
(Czapinski and Barnes, 2011)	FSSP	TS	low-level	1C/RS/SPSS	master-slave	GPGPU	[<=89]
(James et al., 2009)	AP	TS	coop. multi-search	pC/C/MPDS	communication via memory	shared memory	sublinear (n=10)

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Czapiński, 2013)	AP	TS	1. low-level, 2. coop. multi-search	1C/RS/SPSS	master-slave	GPGPU	[<=420]
(Bukata et al., 2015)	Other	TS	coop. multi-search	pC/KS/MPSS	n/a	1. threads, 2. GPGPU	1. linear (n=4), 2. [7.12-50.4]
(Cordeau and Maischberger, 2012)	VRP	TS	low-level	1C/RS/SPSS	n/a	message passing	sublinear (n=10-80)
(Wei et al., 2017)	FSSP	TS	low-level	1C/RS/SPSS	n/a	GPGPU	[1.15 -139.87]
(Janiak et al., 2008)	TSP,FSSP	TS	low-level	1C/RS/SPSS	master-slave	GPGPU	sublinear (n=4)
(Shylo et al., 2011)	JSSP	TS	indep. multi-search	pC/?/?	n/a	n/a	two algorithm versions: 1. superlinear (n<2-20), 2.linear (n=2-20)
(Jin et al., 2014)	VRP	TS	coop. multi-search	pC/KC/MPDS	master-slave	message passing	n/a
(Bożejko et al., 2009)	JSSP	SA	low-level	1C/RS/SPSS	master-slave	SIMD	[1.7-5.6]
(Bożejko et al., 2016)	FSSP	TS	low-level	1C/RS/SPSS	n/a	message passing	sublinear (n=2-10) (no information on which algorithm was analyzed)
(Bożejko et al., 2016)	FSSP	SA	low-level	1C/RS/SPSS	n/a	message passing	sublinear (n=2-10) (no information on which algorithm was analyzed)
(Caniou et al., 2012)	Other	GRAS	indep. multi-search	pC/RS/MP?S	n/a	n/a	sublinear/linear (n=32-256]
(Hifi et al., 2014)	KP	OSSH	domain decomposition	pC/KC/SPDS	master-slave	message passing	n/a
(Jin et al., 2011)	VRP	TS	coop. multi-search	pC/C/MPDS	master-slave	message passing	n/a
(Lazarova and Borovska, 2008)	TSP	SA	1. coop. multi-search, 2. ?, 3. coop. multi-search	1. pC/?/MPDS, 2. ?, 3. pC/RS/MPDS	master-slave	hybrid (message passing + shared memory)	results vary between instances (n=2-10)
(Maischberger and Cordeau, 2011)	VRP	TS	low-level	1C/RS/SPSS	n/a	n/a	n/a
(Santos et al., 2010)	Other	GRAS	domain decomposition	1C/RS/SPSS	master-slave	GPGPU	[1.14-13.89]
(Van Luong et al., 2013)	1. AP, 2. BFP, 3. TSP	TS	low-level	1C/RS/SPSS	master-slave	GPGPU	1. [0.5-18.6], 2. [39.2-243], 3. [0.6-19.9]
(Aydin and Sevkli, 2008)	JSSP	VNS	coop. multi-search	1. pC/RS/SPDS, 2.pC/C/MPDS, 3.pC/C/MPDS, 4.pC/C/MPDS	1. master-slave, 2. master-slave, 3. ring, 4. mesh	message passing	n/a
(Dai et al., 2009)	Other	TS	1. low-level, 2. domain decomposition	1. 1C/RS/SPSS, 2. pC/C/MPSS	1. master-slave, 2. n/a	n/a	1. sublinear, 2. n/a
(Melab et al., 2011)	AP	TS	low-level	1C/RS/SPSS	master-slave	GPGPU	[0.9-15.1]
(Polacek et al., 2008)	VRP	VNS	coop. multi-search	pC/C/MSDS	master-slave	message passing	linear (n=2,...,32)

Table B.7: Computational parallelization studies (single-solution based metaheuristics)

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Yu et al., 2011b)	VRP	ACO	coop. multi-search	pC/RS/MP?S	ring	n/a	n/a
(Ling et al., 2012)	TSP	ACO	coop. multi-search	pC/C/MP?S	communication partners selected dynamically	message passing	sublinear (n=5-35)
(Cecilia et al., 2013)	TSP	ACO	low-level	1C/RS/SPSS	mesh	GPGPU	[0.6-22]
(Delevacq et al., 2013)	TSP	ACO	low-level	1C/RS/SPSS	master-slave	GPGPU	a. [2-19.47], b. [0.06-23.6]
(Zhou et al., 2017)	TSP	ACO	low-level	1C/RS/SPSS	n/a	hybrid (shared memory (task-based) + SIMD)	linear (n=2-16)
(Hadian et al., 2012)	TSP	ACO	low-level	1C/RS/SPSS	master-slave	threads	sublinear (n=2-24)
(Yang et al., 2016)	TSP	ACO	coop. multi-search	pC/RS/MPDS	hybrid (master-slave + fully connected mesh)	message passing	sublinear (n=2-8)
(Cecilia et al., 2011)	TSP	ACO	low-level	1C/RS/SPSS	master-slave	GPGPU	n/a
(Skinderowicz, 2016)	TSP	ACO	low-level	1C/RS/SPSS	master-slave	GPGPU	[5-25]
(Abouelfarag et al., 2015)	TSP	ACO	low-level	1C/RS/SPSS	master-slave	shared memory	sublinear (n=2-32)
(Luo et al., 2014)	BFP	BCO	coop. multi-search	pC/RS/MPDS	several fully connected meshs (isolated from each other)	GPGPU	[13-56]
(Aitzai and Boudhar, 2013)	JSSP	PSO	domain decomposition	1C/RS/MPSS	master-slave	message passing	n/a
(Massobrio et al., 2017)	Other	GA	low-level	1C/RS/SPSS	master-slave	threads	n/a
(Fabris and Krohling, 2012)	Other	OEA	low-level	1C/RS/SPSS	master-slave	GPGPU	[1.3-14]
(Liu et al., 2016)	Other	GA	coop. multi-search	pC/RS/MPDS	ring	message passing	n/a
(Yu et al., 2017)	Other	PSO	hybrid (low-level + coop. multi-search)	pC/?/MPDS	hybrid (master-slave + ring)	message passing	[3-18] (n=5)
(Pedroso et al., 2017)	Other	OEA	coop. multi-search	pC/RS/MPSS	master-slave	n/a	superlinear (n<=14)
(Cao et al., 2017)	Other	OEA	coop. multi-search	pC/RS/MPSS	grid	message passing	sublinear (n=360)
(Dorronsoro et al., 2013)	MSP	GA	coop. multi-search	pC/RS/MPDS	fully connected mash	threads	varies between algorithms and instances (n=4-8)
(Dorronsoro et al., 2013)	MSP	OEA	coop. multi-search	pC/RS/MPDS	fully connected mash	threads	varies between algorithms and instances (n=4-8)
(Aldinucci et al., 2016)	Other	OEA	low-level	1C/RS/SPSS	master-slave	threads	sublinear/linear (n<=24)

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Figueira et al., 2010)	Other	OEA	indep. multi-search	pC/RS/MPSS	master-slave	message passing	n/a
(Derbel et al., 2014)	Other	OEA	coop. multi-search	pC/?/DP?S	double-linked list	message passing	sublinear/linear (n<=128)
(Defersha and Chen, 2008)	Other	GA	coop. multi-search	pC/RS/MPDS	1.-2. ring, 3.-4. mesh, 5. fully connected mesh, 6. randomly connected mesh	message passing	n/a
(Defersha and Chen, 2010)	JSSP	GA	coop. multi-search	pC/RS/MPSS	randomly connected mesh	message passing	n/a
(Huang et al., 2012)	FSSP	GA	low-level	1C/RS/SPSS	master-slave	GPGPU	[16.6-19.1]
(Liu and Wang, 2015)	AP	GA	coop. multi-search	pC/RS/MPDS	grid	message passing	n/a
(Defersha and Chen, 2012)	FSSP	GA	coop. multi-search	pC/RS/MPSS	randomly connected mesh	message passing	n/a
(Homberger, 2008)	Other	GA	1. coop. multi-search, 2. indep. multi-search	1. pC/RS/MPSS, 2. pC/RS/MPSS	fully connected mesh	LAN file system	algorithm 1. (with migration): superlinear (n=30), algorithm 2. (independent): n/a
(Gao et al., 2009)	MSP	GA	coop. multi-search	pC/RS/MPSS	master-slave	message passing	n/a
(Tosun et al., 2013)	AP	GA	coop. multi-search	pC/RS/MPSS	master-slave	message passing	linear (n=26-201)
(Zhang et al., 2016)	Other	GA	coop. multi-search	pC/RS/MPDS	master-slave	message passing	n/a
(Lu et al., 2014)	MSP	GA	coop. multi-search	pC/RS/SPDS	n/a	n/a	n/a
(Abu-lebdeh et al., 2016)	Other	GA	coop. multi-search	pC/RS/MPDS	master-slave	n/a	superlinear (n=4-8)
(Kang et al., 2016)	TSP	GA	low-level	1C/RS/SPSS	master-slave	GPGPU	n/a
(He et al., 2010)	Other	GA	coop. multi-search	pC/RS/MPDS	ring	n/a	n/a
(Limmer and Fey, 2017)	BFP	GA	1. coop. multi-search, 2. low-level	1. pC/RS/MPSS 2. 1C/RS/SPSS	hybrid (ring + master-slave)	1. shared memory, 2. hybrid (message passing + GPGPU)	n/a
(Abbasian and Mouhoub, 2013)	GTP	GA	hybrid (low-level + coop. multi-search)	1C/RS/SPSS	master-slave	n/a	varies between instances (n=24)
(Roberge et al., 2013)	Other	GA	coop. multi-search	pC/RS/MPDS	n/a	n/a	linear (n=2-8)
(Roberge et al., 2013)	Other	PSO	n/a	n/a	n/a	n/a	linear (n=2-8)
(Scheerlinck et al., 2012)	Other	PSO	cooperative multi-search	1. pC/RS/MPSS	master-slave	message passing	n/a
(Ze-Shu et al., 2017)	GTP	PSO	low-level	1C/RS/SPSS	master-slave	GPGPU	n/a
(Qu et al., 2017)	GTP	PSO	low-level	1C/RS/SPSS	master-slave	GPGPU	n/a
(Hung and Wang, 2012)	Other	PSO	low-level	1C/RS/SPSS	communication via memory	GPGPU	[<=281.7]

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Laguna-Sanchez et al., 2009)	BFP	PSO	1. low-level, 2. low-level, 3. n/a	1. 1C/RS/SPSS, 2. 1C/RS/SPSS, 3. n/a	1. and 2. master-slave, 3. communication via memory	GPGPU	[1-28]
(Mussi et al., 2011)	Other	PSO	1. domain decomposition, 2. coop. multi-search	1. 1C/RS/MP?S, 2. pC/?/?	1. communication via memory, 2. master-slave	GPGPU	[<45]
(Kerkhove and Vanhoucke, 2017)	Other	SSPR	1. low-level, 2. indep. multi-search	1. 1C/RS/SPSS, 2. pC/RS/MPDS	n/a	n/a	n/a
(Božejko, 2009)	FSSP	SSPR	1. low-level, 2. indep. multi-search	1. 1C/RS/SPSS, 2. pC/RS/?	master-slave	message passing	linear/superlinear (n=2-16)
(Baños et al., 2014)	VRP	OEA	coop. multi-search	pc/C/MPDS	master-slave	1. message passing, 2. shared memory, 3. hybrid (message passing + shared memory)	1. sublinear (n=4), 2. sublinear (m=4), 3 sublinear (n=2 nodes with m=2 threads each)
(Deep et al., 2010)	BFP	PSO	coop. multi-search	1. pC/RS/MPSS, 2. pC/RS/MPSS, 3. pC/RS/MPSS	master-slave	message passing	(type and development of slope over n) of speedup substantially varies between benchmark functions (n=2-12)
(Diego et al., 2012)	VRP	ACO	n/a	n/a	n/a	GPGPU	[<13]
(Ding et al., 2013)	BFP	FA	coop. multi-search	pC/RS/MPDS	n/a	GPGPU	[59.2- 190]
(Ding et al., 2013)	BFP	PSO	coop. multi-search	pC/RS/MPDS	n/a	GPGPU	[59.2- 190]
(Dongdong et al., 2010)	AP	ACO	1. low-level, 2. indep. multi-search	1. 1C/RS/SPSS, 2. pC/RS/MPSS	1. master-slave, 2. n/a	1. shared memory, 2. threads	1. linear/sublinear (n=1-32), 2. sublinear (n=1-32)
(Lančinskas and Žilinskas, 2013)	FLP	GA	coop. multi-search	pC/RS/MPDS	master-slave	message passing	linear/sublinear (n=128-1024)
(Lančinskas and Žilinskas, 2012)	FLP	GA	coop. multi-search	pC/RS/MPDS	master-slave	1. message passing, 2. hybrid (message passing + shared memory)	1. linear/sublinear (n=32-512), sublinear (n=1024-2048), 2. sublinear (n= 32-512 nodes with m=4 threads each), sublinear (n= 8-128 nodes with m=16 threads each)
(Lazarova and Borovska, 2008)	TSP	ACO	1. coop. multi-search, 2. ?, 3. coop. multi-search	1. pC/?/MPDS, 2. ?, 3. pC/RS/MPDS	master-slave	hybrid (message passing + shared memory)	results vary between instances (n=2-10)
(Lazarova and Borovska, 2008)	TSP	GA	1. coop. mutli-search, 2. ?, 3. coop. multi-search	1. pC/?/MPDS, 2. ?, 3. pC/RS/MPDS	ring	hybrid (message passing + shared memory)	sublinear (n=2-10)
(Nebro and Durillo, 2010)	BFP	OEA	coop. multi-search	pC/RS/MPDS	n/a	threads	machine 1: linear (n=1-2), sublinear (n=4-32), machine 2: linear (n=1-32)

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Nowotniak and Kucharski, 2011)	KP	OEA	coop. multi-search	pC/RS/MPDS	n/a	GPGPU	[120 - 400]
(Redondo et al., 2008)	FLP	OEA	coop. multi-search	pC/RS/MPDS	1. ring, 2. master-slave, 3. hybrid (ring + master-slave)	message passing	1. linear (n=2-8), sublinear (n=16-32), 2. n/a, 3. n/a., 4. linear/super-linear (n=2), linear (n=4-8), linear/sub-linear (n=16), sub-linear (n=32)
(Sancı and İşler, 2011)	Other	GA	1.low-level, 2. cooperative multi-search	1. 1C/RS/SPSS, 2. pC/RS/MPDS	n/a	GPGPU	[16.3-20.1]
(Tsutsui, 2008)	AP	ACO	1. low-level, 2. indep. multi-search	1. 1C/RS/SPSS; 2. pC/C/?	1.master-slave, 2a. fully connected mesh, 2b. other (replace-worst), 2c. other (unconnected)	threads	1. sublinear (n=4; further, unspecified n), 2. sublinear (n=4)
(Umbarkar et al., 2014)	BFP	GA	low-level	1C/RS/SPSS	fully connected mesh	threads	n/a
(Wang et al., 2008)	FLP	PSO	domain decomposition	pC/KC/MPSS	master-slave	shared memory	sublinear (n=2)
(Wang et al., 2012)	Other	GA	low-level	1C/RS/SPSS	master-slave	GPGPU	[23-32.86]
(Weber et al., 2011)	n/a	OEA	coop. multi-search	pC/RS/MPDS	n/a	n/a	n/a
(You, 2009)	TSP	ACO	indep. multi-search	pC/?/MP?S	n/a	GPGPU	[<22]
(Zhao et al., 2011)	TSP	GA	coop. multi-search	pC/?/MPDS	n/a	GPGPU	[3.3-5.3]
(Zhao et al., 2011)	TSP	ACO	coop. multi-search	pC/?/MPDS	n/a	GPGPU	[2.9-8.4]
(Zhao et al., 2011)	TSP	OEA	coop. multi-search	pC/?/MPDS	n/a	GPGPU	[3.15-15.8]
(Davidovic et al., 2011)	MSP	BCO	1. low-level, 2. indep. multi-search, 3. indep. multi-search	1.1C/RS/SPSS, 2. pC/RS/SPDS, 3. pC/RS/SPDS	message passing	MPI	1. sublinear (n=2-12), 2. linear (n=2-5), 3. superlinear (n=2-12)
(Subotic et al., 2011)	BFP	BCO	1. indep. multi-search, 2. coop. multi-search, 3. coop. multi-search	1. pC/RS/?SDS, 2. pC/RS/?SDS, 3. pC/RS/?SDS	threads	Java	1. sublinear (n=4), 2. n/a, 3. n/a
(Izzo et al., 2009)	BFP	OEA	coop. multi-search	pC/C/MPDS	threads	C++, POSIX threads	n/a
(Vallada and Ruiz, 2009)	FSSP	GA	coop. multi-search	pC/C/?PDS	message passing	Delphi 2006, Msgconnect	n/a
(Arellano-Verdejo et al., 2017)	BFP	GA	coop. multi-search	pC/RS/MP?S	master-slave	n/a	n/a

Table B.8: Computational parallelization studies (population-based metaheuristics)

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Thiruvady et al., 2016)	JSSP	HM (a. ACO + local search, b. ACO + SA)	1a. low-level, 1b. coop. multi-search, 2. coop. multi-search	1a. 1C/RS/SPSS, 1b. pC/C/?PDS, 2. pC/C/SPDS	n/a	shared memory	n/a
(Delevacq et al., 2013)	TSP	HM (ACO + local search)	low-level	1C/RS/SPSS	master-slave	GPGPU	a. [0.17-8.03], b. [0.06-23.6]
(Arrondo et al., 2014)	Other	HM (OEA + local search)	1. low-level, 2. low-level, 3. coop. multi-search	1. 1C/RS/SPSS, 2. 1C/RS/SPSS, 3. pC/?/MP?S	master-slave	1. message passing, 2.shared memory, 3. hybrid (message passing + shared memory)	1. linear (n=2-8), sublinear (n=16-64), 2. linear (n=2-8), 3. sublinear (n=16-64)
(Patvardhan et al., 2016)	KP	HM (OEA + local search)	low-level	1C/RS/SPSS	master-slave	shared memory	varies between instances (n=24)
(Nesmachnow et al., 2012)	MSP	HM (OEA + local search)	coop. multi-search	pC/RS/MPDS	ring	hybrid (shared memory + message passing)	n/a
(Redondo et al., 2011)	Other	HM (GA + local search)	low-level	1C/RS/SPSS	master-slave	threads	sublinear/linear (n=2-16)
(Mezmaz et al., 2011)	MSP	HM (GA + scheduling heuristic)	coop. multi-search	pC/C/MPDS	master-slave	n/a	sublinear (n=8-64)
(Ku et al., 2011)	Other	HM (GA + SA)	coop. multi-search	pC/RS/MPDS	tree	n/a	n/a
(Li et al., 2017)	TSP	HM (GA + SA)	low-level	1C/RS/SPSS	master-slave	GPGPU	n/a
(Yu et al., 2011a)	Other	HM (GA + TS)	coop. multi-search	pC/RS/MPDS	ring	message passing	n/a
(Munawar et al., 2009)	Other	HM (GA + hill climbing)	hybrid (low-level + coop. multi-search)	n/a	grid	GPGPU	[16-25]

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Ravetti et al., 2012)	FSSP	HM (GA + local search + greedy algorithms)	cooperative multi-search	pC/C/MPDS	fully-connected mesh	shared memory	n/a
(Ben Mabrouk et al., 2009)	GTP	HM (GA + TS)	indep. multi-search	pC/RS/MPDS	master-slave	message passing	linear/sublinear (n=2-24)
(Subramanian et al., 2010)	VRP	HM (VNS + iterated local search)	low-level	1C/RS/SPSS	master-slave	message passing	sublinear (n=2-256)
(Scheerlinck et al., 2012)	Other	HM (PSO + gradient descent)	cooperative multi-search	pC/RS/MPSS	master-slave	message passing	n/a
(Czapinski, 2010)	FSSP	HM (cluster algorithm for simulated annealing + GA)	domain decomposition	1C/RS/MPDS	master-slave	message passing	1. linear (n=2-64), 2. n/a
(Banos et al., 2013)	VRP	HM (GA + SA)	coop. multi-search	pC/RS/MPDS	master-slave	message passing	sublinear (n=2-4)
(Olensek et al., 2011)	BFP	HM (SA + differential evolution)	low-level	1C/RS/SPSS	master-slave	message passing	linear (n=2-8)
(Fujimoto and Tsutsui, 2011)	TSP	HM (OEA + 2-opt local search)	low-level	1C/RS/SPSS	n/a	GPGPU	[9.3-24.2]
(Ibri et al., 2010)	Other	HM (ACO + TS)	1. indep. multi-search, 2. domain decomposition	1. pC/C/?, 2. 1C/RS/?	master-slave	threads	sublinear (n=2-30)
(Lančinskas and Žilinskas, 2013)	FLP	HM (LS + GA)	coop. multi-search	pC/RS/MPDS	master-slave	hybrid (message passing +shared memory)	linear/sublinear (n=128-1024) (only algorithm version 2)
(Van Luong et al., 2012)	AP	HM (algorithms unspecified)	domain decomposition	pC/RS/MPSS	n/a	1. threads, 2. GPGPU, 3. hybrid (threads + GPGPU)	1. sublinear (n=4-8), 2. [7.0-14.4], 3. [9.8-16.7]

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Xhafa and Duran, 2008)	MSP	HM (algorithms unspecified)	1. indep. multi-search, 2. low.level, 3. n/a	1. pC/?/MPDS, 2. 1C/RS/SPSS, 3. n/a	1. other (unconnected), 2. master-slave, 3. other (two-level: grid + mesh)	message passing	sublinear (n=3-9)
(Zhao et al., 2011)	TSP	HM	coop. multi-search	pC/?/MPDS	n/a	GPGPU	[2.8-8.3]
(Zhu and Curry, 2009)	BFP	HM (ACO + pattern search)	low-level	1C/RS/SPSS	master-slave	GPGPU	[66.85-403.91]

Table B.9: Computational parallelization studies (hybrid metaheuristics)

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Stanojevic et al., 2015)	FLP	MH (B-a-X + OEA)	domain decomposition	pC/RS/MPSS	n/a	shared memory	n/a
(Ozden et al., 2017)	TSP	PSH	domain decomposition	1C/RS/MPSS	master-slave	message passing	n/a
(Groer et al., 2011)	VRP	MH (PSEA + OSSH)	coop. multi-search	pC/KC/MPDS	master-slave	message passing	linear (n=8-64)
(Lahrichi et al., 2015)	VRP	MS (integrative cooperative search method)	hybrid (domain decomposition and cooperative multi-search)	pC/KC/MPDS	communication via memory	shared memory	n/a
(Chaves-Gonzalez et al., 2011)	AP	MS (several meta-heuristics parallelized at the same time)	coop. multi-search	pC/RS/MPDS	master-slave	message passing	n/a
(Koc and Mehrotra, 2017)	MILP	PSH	coop. multi-search	pC/C/MPDS	master-slave	message passing	n/a
(Redondo et al., 2016)	Other	PSH	low-level	1C/RS/SPSS	n/a	threads	linear/sublinear (n=2-8)
(Hemmelmayr, 2015)	Other	PSH	1. domain decomposition, 2. coop. multi-search	1. 1C/KS/SPSS; 2. pC/C/MPDS	1. master-slave, 2. mesh	message passing	1. sublinear (n=4), 2. sublinear (n=7)
(Juan et al., 2013)	VRP	OH (Monte Carlo simulation inside a heuristic-randomization process)	n/a	n/a	n/a	threads	n/a
(Benedicic et al., 2014)	Other	PSH	n/a	n/a	master-slave	GPGPU	n/a
(Benedicic et al., 2014)	Other	OH (method based on autonomous agents)	n/a	n/a	master-slave	GPGPU	n/a
(Gomes et al., 2008)	Other	PSH	low-level	1C/RS/SPSS	master-slave	message passing	n/a

Reference	Problem	Algorithm	Parallelization strategy	Process & search control	Communication topology	Programming model	Scalability
(Lancinskas et al., 2015)	FLP	PSH	low-level	1C/RS/SPSS	master-slave	1. shared memory, 2. message passing	1. linear (n=2-16), 2. linear (n=31-192)
(Baumelt et al., 2016)	Other	PSH	domain decomposition	1C/RS/MPSS	n/a	GPGPU	[2-15]
(Bożejko, 2009)	FSSP	PSH	low-level	1C/RS/SPSS	master-slave	n/a	[2-3] (unreported n)
(Dobrian et al., 2011)	GTP	PSH	domain decomposition	pC/RS/MPSS	ring	message passing	n/a
(Ismail et al., 2011)	TSP	PSH	n/a	n/a	master-slave	shared memory (task-oriented, implemented as threads)	sublinear (n=2)
(Luo et al., 2015)	Other	PSH	low-level	1C/RS/SPSS	master-slave	threads	n/a
(Sancı and İşler, 2011)	Other	OH	1.low-level, 2. coop. multi-search	1. 1C/RS/SPSS, 2. pC/RS/MPDS	n/a	GPGPU	[2.4-21.49]
(Sathe et al., 2012)	GTP	OH	domain decomposition	pC/RS/MPSS	fully connected mesh	hybrid (message passing + shared memory)	relative speedup: sublinear/linear (n=2-16), sublinear (n=32-1024); weak speedup: mixed results (n=1-1024)
(Vidal et al., 2017)	Other	MS (systolic neighborhood search + GA)	coop. multi-search	pC/RS/MPDS	master-slave	hybrid (shared memory + GPGPU)	n/a

Table B.10: Computational parallelization studies (problem-specific heuristics, other heuristics, matheuristics, and multi-search algorithms:)

Appendix C. References of Appendix

Bozejko, W., 2012. On single-walk parallelization of the job shop problem solving algorithms. Computers & Operations Research 39, 2258–2264.

German Academic Association for Business Research (VHB), . VHB-JOURQUAL3. http://vhbonline.org/vhb4you/jourqual/vhb-jourqual-3/teilrating-or/.

Östermark, R., 2014. Solving difficult mixed integer and disjunctive non-linear problems on single and parallel processors. Applied Soft Computing 24, 385–405.

Östermark, R., 2015. A parallel algorithm for optimizing the capital structure contingent on maximum value at risk. Kybernetes 44, 384–405.