ScatterD: Spatial Deployment Optimization with Hybrid Heuristic / Evolutionary Algorithms

Jules White, Brian Dougherty, Chris Thompson, and Douglas C. Schmidt
Department of Electrical Engineering and Computer Science,
Vanderbilt University, Nashville, TN, USA
{jules,briand,schmidt}@dre.vanderbilt.edu & chris.m.thompson@vanderbilt.edu

Abstract

Distributed real-time and embedded (DRE) systems can be composed of hundreds of software components running across tens or hundreds of networked processors that are physically separated from one another. A key concern in DRE systems is determining the spatial deployment topology, which is how the software components map to the underlying hardware components. Optimizations, such as placing software components with high-frequency communications on processors that are closer together, can yield a number of important benefits, such as reduced power consumption due to decreased wireless transmission power required to communicate between the processing nodes.

Determining a spatial deployment plan across a series of processors that will minimize power consumption is hard since the spatial deployment plan must respect a combination of real-time scheduling, fault-tolerance, resource, and other complex constraints. This paper presents a hybrid heuristic/evolutionary algorithm, called ScatterD, for automatically generating spatial deployment plans that minimize power consumption. This work provides the following contributions to the study of spatial deployment optimization for power consumption minimization: (1) it combines heuristic bin-packing with an evolutionary algorithm to produce a hybrid algorithm with excellent deployment derivation capabilities and scalability, (2) it shows how a unique representation of the spatial deployment solution space integrates the heuristic and evolutionary algorithms, and (3) it analyzes the results of experiments performed with data derived from a large-scale avionics system that compares ScatterD with other automated deployment techniques. These results show that ScatterD reduces power consumption by between 6% and 240% more than standard bin-packing, genetic, and particle swarm optimization algorithms.

1 Introduction

Modern mobile sensor networks, cooperative robot swarms, and fractionated spacecraft are examples of distributed real-time and embedded (DRE) systems. A sensor platform can have 10's or 100's of processing units, spatially separated on several physical platforms, and connected through multiple ad-hoc network links between the processors. Moreover, several hundred software components can be distributed across these multiple networked processors.

The software in these types of DRE systems has traditionally been tightly-coupled to the underlying hardware. In satellites, for example, each software component is often designed to only function with the APIs provided by a specific hardware setup on a single satellite. The deployment topology of the software components (*e.g.*, how software is mapped to hardware processors) is thus dictated by this tight-coupling.

Recent trends in DRE development have begun decoupling the software from the underlying hardware. For example, so-called fractionated spacecraft [8] are being designed to operate in groups and provide a common runtime environment for satellite software [30]. The standardized runtime interfaces allow migration of software functionality between satellites with minimal effort to reconfigure and adapt the capabilities of the satellites for different objectives. The same software can be deployed to a group of satellites in multiple unique spatial deployment topologies.

The increased flexibility in deployment topologies allows designers to exploit spatial characteristics of the physical platform to optimize the power consumption of the overall system. For example, two software components that communicate at a high rate can be spatially co-located on the same computing node to avoid communicating through a power consumptive wireless communication link. Moreover, variations in performance per watt between processors or cores can be exploited by the deployment topology to use processing nodes that consume less power.

Effective spatial deployment of distributed systems can also lower power consumption simply by more effectively utilizing hardware. By packing software more tightly onto

hardware, fewer hardware resources and less power can be used. For example, roughly four pounds of cooling, power supply, and other supporting hardware are needed for each pound of processing hardware in a plane. Reducing hardware not only decreases DRE system cost, but also facilitates increased ranges for planes/cars, decreased fuel and power consumption, and reduced waste. Each pound of processor savings on a plane has been estimated to decrease fuel costs by $200 and decrease greenhouse gas production from less burned fuel [47].

Determining how to optimize the spatial deployment of software to hardware in DRE systems to minimize power consumption is hard [6, 10] due to the large number of complex constraints that must be satisfied [28, 34, 40, 38, 23, 7]. Developers must ensure that each software component receives sufficient processing time to meet any real-time scheduling constraints [25]. Resource constraints, such as total available memory on each processor, must also be respected by the spatial mapping of software to hardware [46, 13]. Finally, components can have complex placement or co-location constraints, such as requiring specific software components to be deployed to processors a minimum distance from the perimeter of a vehicle or robot to provide crash survivability [13].

Open problem. A key challenge for DRE system developers is that the techniques for determining the best way to deploy software components onto hardware to optimize power consumption have not kept pace with the decoupling efforts. Manual techniques for determining spatial deployment topologies—which are commonly used by DRE system developers—do not scale well and produce solutions that consume more power and hardware than is necessary. Assigning real-time tasks in both multi-processor and single-processor machines has been shown to be NP-Hard [9]. Some techniques are available for scheduling software on multi-core systems on a chip, but these techiques are not designed for optimizing deployment across multiple chips that are spatially separated.

Research using integer programming and constraint programming has been performed on optimizing power consumption and performance of software components mapped to multi-core chips [28, 34, 40, 38, 23, 7]. This prior work, however, has focused on scenarios where all cores are physically located on the same chip. A research gap therefore exists for techniques that can perform deployment optimization across cores on multiple chips, distributed across multiple networked platforms, and spatially separated from one another.

Solution approach → Hybrid heuristic/evolutionary algorithms. Existing work on bin-packing algorithms for multi-processor scheduling and evolutionary algorithms for non-linear optimization have addressed specific points in the spatial deployment problem space. The chief problem is that a comprehensive approach for adapting and using the disparate techniques together has not been developed. To address this problem, we present a spatial deployment algorithm, called the *Scatter Deployment Algorithm* (ScatterD), which is a hybrid deployment solver that can optimize for power consumption. ScatterD can adhere to a combination of deployment constraints that cannot be satisfied by a single technique. ScatterD also leverages a novel representation of the deployment design space based on input permutations to a bin-packing algorithm, as discussed in Section 3.4.

This paper provides the following contributions to the study of automated spatial deployment techniques for minimizing power consumption:

- We describe how ScatterD's hybrid heuristic/evolutionary algorithm optimizes spatial deployment topologies by utilizing variations in network link and processing node power consumption to minimize overall system power requirements,
- We show ScatterD's novel representation of deployment design spaces based on input permutations to a bin-packing algorithm can be used to combine a heuristic bin-packing with an evolutionary algorithm and overcome many limitations of applying either independently,
- We show how evolutionary algorithms using this modified design space representation can be seeded with the output from a heuristic bin-packer to improve search convergence speed,
- We present empirical results demonstrating the improvements in power consumption yielded by the ScatterD deployment algorithm versus heuristic bin-packing algorithms and evolutionary algorithms, and
- We present empirical results that show that the ScatterD algorithm is significantly more scalable than a constraint satisfaction programming (CSP) approach.

The remainder of this paper is organized as follows: Section 2 presents a case study used throughout the paper to motivate the challenges of devising a complex spatial deployment plan for fractionated spacecraft deployment; Section 3 presents the ScatterD hybrid heuristic/evolutionary deployment algorithm; Section 4 analyzes the results from experiments with ScatterD; Section 5 compares ScatterD with related work on deployment automation; and Section 6 presents concluding remarks and lessons learned.

2 A Fractionated Spacecraft Case Study

This section presents a case study of a weather satellite designed using a fractionated spacecraft architecture [8].

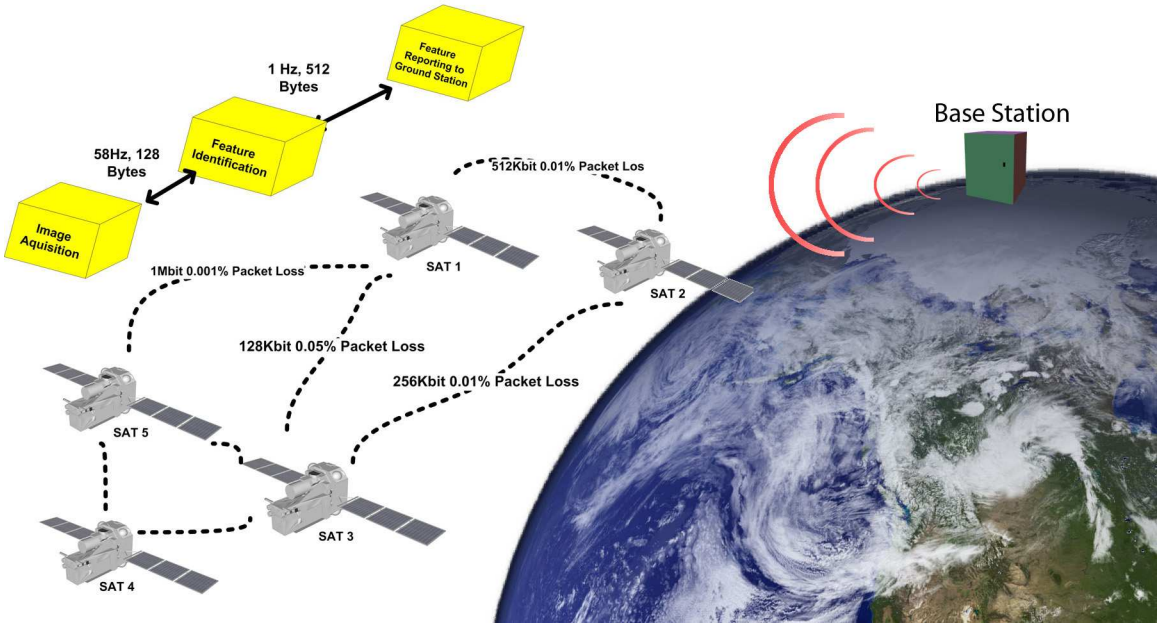


Figure 1. A Fractionated Spacecraft Spatial Deployment Problem

We use this case study throughout the paper to motivate the challenges of devising complex spatial deployment plans.

2.1 Overview of Fractionated Spacecraft

The fractionated spacecraft weather satellite is controlled by a ground station. The satellite runs a set of software components (such as `Image Acquisition` for capturing weather images, `Feature Identification` for identifying important characteristics of the images, and `Feature Reporting` for relaying image information to a ground station) that is dictated by the ground control station. The ground control station can also dynamically change the spatial deployment of the software components to satellites to improve performance, change power consumption characteristics, or adapt to hardware failures.

The example fractionated spacecraft deployment problem in Figure 1 shows how developers must determine a deployment plan that maps the three imaging software components in the upper left hand to the group of satellites. The generated deployment topology must ensure that each software component meets its real-time deadlines, memory consumption does not exceed a maximum limit, communication links provide sufficient bandwidth and connectivity for processing, and as little power as possible is used.

We use this fractionated spacecraft scenario to showcase several key challenges in Section 2.2. The example has been reduced in scale to simplify the process of illustrating the complexities of DRE system deployment. Production deployment problems involve hundreds of software components and thousands of messaging interactions, as opposed to the three shown in this example.

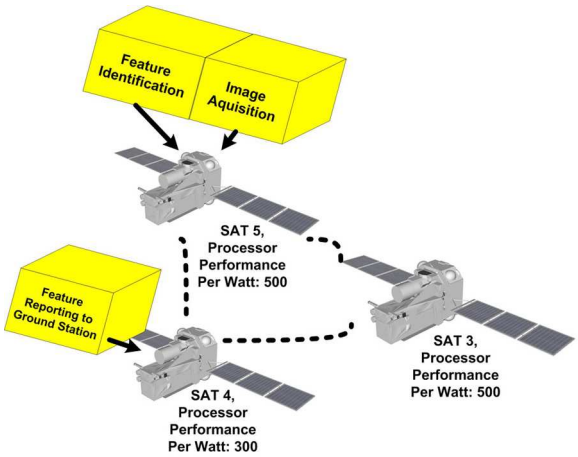
2.2 Spatial Deployment Challenges for DRE Systems

A deployment topology is a mapping of software components and their associated tasks to a hardware processor components. In real-time systems, either fixed priority scheduling algorithms, such as rate-monotonic (RM) scheduling [48], or dynamic priority scheduling algorithms, such as earliest-deadline-first (EDF) [48], control the execution ordering of the individual tasks on the processors. A fundamental constraint of finding a spatial deployment topology for DRE systems is that the topology must ensure that no processor is assigned more tasks than it can support. Finding a deployment topology for a series of software components that ensures schedulability of all tasks has been shown to be NP-Hard [9].

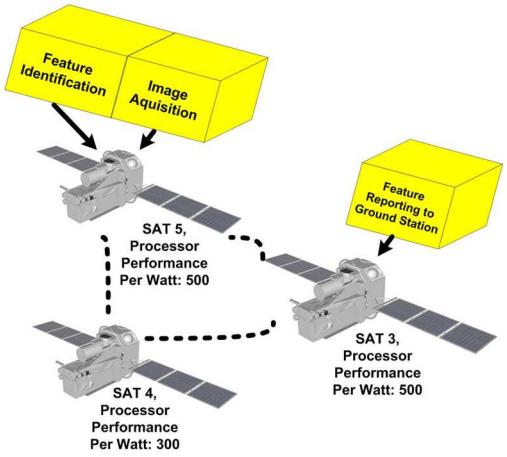
Since the problem is NP-Hard, developers must make various assumptions (such as estimates of the worst-case execution time of software tasks) and employ a variety of heuristic deployment algorithms. A number of algorithms assume processor homogeneity and use variations of heuristic bin-packing algorithms [16, 14, 35, 15, 9]. For example, rate-monotonic first fit scheduling allocates components to processors using a first fit bin-packing algorithm.

Although these algorithms help facilitate automated generation of deployment topologies for real-time systems, they do not account for how variations in deployment topol-

ogy affect the power consumption of a DRE system. Network links—particularly wireless ad-hoc network links—vary in power consumption based on the transmission power required to send a signal across the space between two processing nodes. Moreover, hardware node variations in clock speed, voltage, and other attributes can impact the performance per watt that they provide. It is critical to account for these variations in DRE systems when designing a spatial deployment topology that optimizes for power consumption. The remainder of this section explores key challenges of using existing deployment algorithms to optimize the deployment topology of a DRE system for power consumption.



(a) A Software Deployment Topology for the Fractionated Spacecraft



(b) A Software Deployment Topology for the Fractionated Spacecraft with Superior Performance per Watt

Figure 2. Comparison of Two Spatial Deployment Topologies for Processor Power Consumption

2.3 Challenge 1: Accounting for Variations in Performance Per Watt of Processing Cores

Current deployment techniques, such as bin-packing, are designed to minimize the number of computational nodes that are utilized by a deployment topology, but do not provide assurance or optimization of the power consumed by the deployment topology. The goal of solving a bin-packing problem is to take a set of items with varying sizes and pack them into a series of limited size bins using as few bins as possible. Properties of the power consumption of the bins are not considered in the problem. In many spatial deployment domains, such as the fractionated spacecraft imaging platform shown in Figure 1, power is severely limited and minimizing its consumption is a key concern.

Heuristic bin-packing algorithms [16, 14, 35, 15, 9]. typically use an incremental algorithm that sorts items and bins based on their sizes and then use a packing strategy, such as first-fit, to allocate each item to a bin. Specialized variations of bin-packing [16, 14, 35, 15, 9] are used to pack tasks onto processors while ensuring real-time schedulability.

For example, both deployment topologies in Figures 2(a) & 2(b) yield a solution that consumes two bins (*e.g.*, satellites). A bin-packing algorithm would consider both solutions to be equally valid–both use exactly two processing cores. Clearly, however, the solution in Figures 2(b) is superior since it uses `Sat 5` and `Sat 3`, which have a higher performance per watt score. A bin-packing algorithm cannot discern the advantage of using `Sat 3` and `Sat 5` compared to `Sat 4` and `Sat 5`. The method that ScatterD uses to overcome this challenge is discussed in Section 3.5.

2.4 Challenge 2: Handling Variations in Network Link Power Consumption

A key determinant of power consumption in DRE systems with mobile ad-hoc networks is radio transmission for wireless network links [20]. The power consumption of a wireless link is directly related to the transmit power required to propagate the transmission over the physical area separating the computational nodes and the frequency that the network link is in use. The spatial deployment topology determines how often network communication is performed, how far apart the nodes are that are involved in the communication, and the types of wireless communication techniques required.

For example, the radio communication link in Figure 1 between `Sat 1` and `Sat 3` spans a shorter distance than the link from `Sat 2` to `Sat 3` and thus consumes less power. Clearly, if the `Image Acquisition` and `Feature Identification` components communicate across one of these links, deploying the components to `Sat 1` and `Sat 3` and using their radio link will con-

sume less power. It is critical that a spatial deployment technique account for these types of variation in power consumption between spatial deployment topologies. Again, work has been done on using heuristic algorithms for task deployment [16, 14, 35, 15, 9] and integer programming for optimizing scheduling [11, 5, 39], but no approach currently optimizes the spatial deployment topology for power consumption concerns.

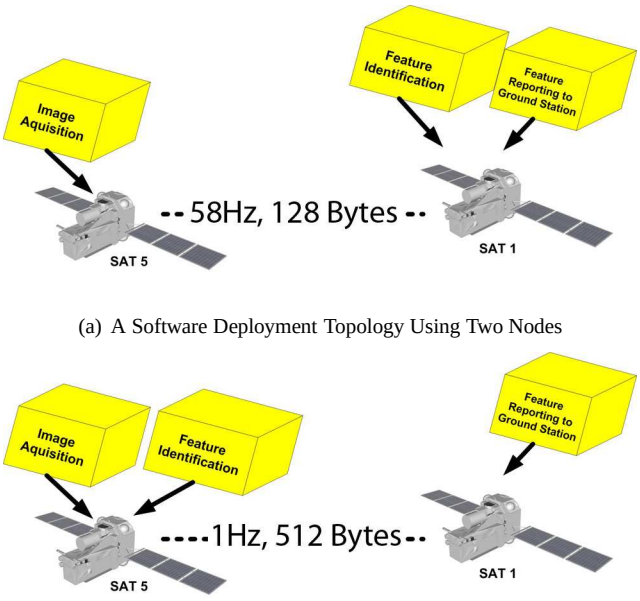
Research using integer programming and constraint programming has been performed on optimizing power consumption on multi-core chips [28, 34, 40, 38, 23, 7]. This prior work, however, has focused on scenarios where all of the cores are physically located on the same chip. When the processing nodes are not physically co-located one the same chip or on the same physical platform, expensive wireless communications must be performed. Since these techniques do not account for the power consumption of the network links, they may not produce an ideal deployment topology.

For example, each satellite in Figure 1 has sufficient CPU capacity to schedule the real-time tasks associated with both the `Image Acquisition` and `Feature Ident` or `Feature Ident` and `Feature Reporting`. Scheduling the tasks of all three software components on one satellite's processor is not possible. As shown in Figures 3(a) & 3(b), from the point of view of a multi-core power consumption algorithm, grouping either set of two software components onto the same satellite is equivalent–both will yield a solution that utilizes two processing nodes with identical power consumption. Clearly, however, spatially co-locating the `Image Acquisition` and `Feature Ident` software components, as shown in Figure 3(b), is a better solution since they communicate at 58hz and consume more power communicating between each other than the `Feature Ident` and `Feature Reporting` sotware components. Colocating the tasks in the same space saves that power from being consumed through network communication. Section 3.5 discusses how ScatterD uses an evolutionary algorithm to drive a heuristic bin-packer to explore multiple solutions with differing network power consumption characteristics.

2.5 Challenge 3: Deployment Automation Technique Scalability

A fundamental constraint of deriving deployment topologies for DRE systems is that all software components must meet their real-time deadlines on the processors to which they are deployed. This basic constraint is known as the *multi-processor scheduling problem* and has been shown to be NP-Hard [9]. Since the problem is NP-Hard, many optimization algorithms with exponential time-complexity are hard to apply.

A number of optimization techniques are available for



(a) A Software Deployment Topology Using Two Nodes

(b) A Software Deployment Topology Using Two Nodes and Less Bandwidth

Figure 3. Comparison of Two Spatial Deployment Topologies Based on Network Utilization

similar linear and non-linear problems. For example, a number of similar deployment problems can be modeled as constraint satisfaction problems (CSPs) [45, 37] or integer programming problems [11, 5, 39]. The drawback with these approaches is that they are based on exponential time algorithms that do not scale up to large spatial deployment problems with 10s of computational nodes, 100s of software components, 1000s of software component communication interactions, and multiple interconnecting networks. Numerous optimization tricks can be applied, such as Bender's Decomposition [24, 22], but research has still only shown the ability to scale to 10's of software components and communication interactions. As discussed in Section 3.3, ScatterD combines an evolutionary algorithm with heuristic bin-packing to avoid the scalability limitations of using an exponential algorithm.

3 The ScatterD Deployment Algorithm

This section describes ScatterD, which is a hybrid heuristic/evolutionary algorithm for optimizing spatial deployment topologies for minimizing power consumption. ScatterD allows DRE system developers to automatically derive deployment topologies that meet a combination of real-time scheduling, memory, co-location, and spatial con-

straints. This section first presents a formal model of spatial deployment for power consumption minimization and then provides an overview of evolutionary algorithms. The section next discusses how ScatterD combines an evolutionary and bin-packing algorithm into a hybrid algorithm and provides an example of how particle swarm optimization (PSO) is turned into a hybrid heuristic/evolutionary algorithm with ScatterD.

3.1 Formal Model of DRE Deployment

To optimize a spatial deployment topology for power consumption, a formal model should first be built that describes the constraints and objective function to optimize. A spatial deployment problem for power consumption minimization can be described as a 7-tuple:

$$Dp =< C, N, s(\vec{T}), r(\vec{T}), p(\vec{T}), co(\vec{T}), e(\vec{T}) >$$

where:

- C is the set of components that must be deployed to the hardware processing nodes
- N is the set of hardware processing nodes
- $\vec{T}$ is a deployment topology described as a vector where the i^{th} position holds the index of the node that i^{th} component is deployed to (*e.g.*, Component 1 to Node 3, $T_1 = 3$)
- $s(\vec{T})$ is a function that returns 0 if the deployment topology assigns components to nodes such that no component will miss any real-time deadlines and otherwise returns the total number of deadlines that will be missed. The precise definition of this function varies based on the scheduling policy, such as rate-monotonic or earliest deadline first, that is used. In this paper, we use a recurrence relation for checking rate-monotonic schedulability [48].
- $r(\vec{T})$ is a function that returns 0 if the deployment topology assigns components to nodes such that no node has its resources, such as RAM, overconsumed and otherwise returns the total number of nodes with overconsumed resources.
- $p(\vec{T})$ is a function that returns 0 if the deployment topology meets spatial attribute-based constraints, such as enusring that components that require a minimum distance from a specific point in the satellite are assigned an appropriate location. Otherwise, the function returns the number of these constraints that are violated.
- $co(\vec{T})$ is a function that returns 0 if the deployment topology assigns components to nodes such that no component co-location constraints, such as requiring two components to be deployed to different nodes, are violated. Otherwise, the number of co-location violations is returned.
- $e(\vec{T})$ is an objective function that calculates the power consumed by a specific deployment topology.

The constraints are described by functions so that they can be adapted to a particular spatial deployment scenario. For example, if the scheduling technique changes, a different function can be provided for $s(\vec{T})$. Similarly, if a set of constraints is not present, a particular function may always return 0.

A correct deployment topology, $\vec{T}$, requires that:

$$|\vec{T}| = |C| \tag{1}$$

$$\forall T_i \subset \vec{T},\ (T_i \geq 0) \wedge (T_i \leq |N| - 1) \tag{2}$$

$$\vec{T} \Rightarrow (co(\vec{T}) + p(\vec{T}) + r(\vec{T}) + s(\vec{T}) = 0) \tag{3}$$

First, as shown in (1), the deployment topology vector, $\vec{T}$, must specify a deployment location for every component. Second, as seen in (2), the node index specified by each position in the deployment vector, $\vec{T}$, must refer to a valid 0-based index in the node array. Finally, in (3), the solution must not to violate any scheduling or other constraints. The goal of an optimized deployment topology is to assure that these correctness constraints are upheld, while minimizing the value of $e(\vec{T})$.

3.2 Evolutionary Algorithms Background

Evolutionary algorithms [4, 21], such as particle swarm optimization (PSO) [29, 42] and genetic programming [32, 33], are a promising technique for generating deployment topologies. With an evolutionary algorithm, a set of randomly generated initial solutions [27] is created and then evolved over a series of generations to reach a final solution. The evolutionary process uses a scoring criteria to identify the best solutions and promote their propagation in the evolution. At the end of the evolutionary process, the highest scoring solution is output as the best result.

Members of a solution population are typically modeled as vectors, where the vector components represent genes or positions of particles. In a PSO algorithm, for example, the vectors represent positions within a multi-dimensional design space. Permuting the vector corresponds to changing the design of the solution.

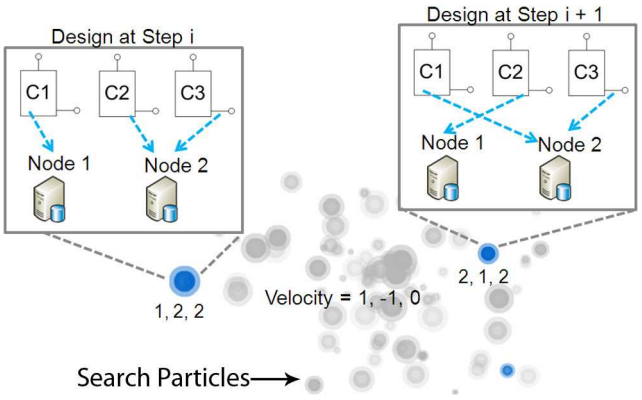


Figure 4. Representing a Spatial Deployment Topology as a Vector

For example, Figure 4 shows an example method of mapping a PSO particle position to a spatial deployment problem. On the left-hand side of Figure 4, the deployment topology vector, $\vec{T} = [1, 2, 2]$ corresponds to a deployment topology where the first software component is deployed to the first hardware node, the second software the to second hardware node, and the third component to the second hardware node. As the PSO evolution proceeds, the design at step i is permuted to the deployment topology $\vec{T} = [2, 1, 2]$ at step i+1, corresponding to the first component being deployed to the second hardware node, the second component to the first hardware node, and the third component to the second hardware node.

The key problem of applying evolutionary algorithms to spatial deployment problems is their poor behavior when the solution space contains large numbers of points in the search space corresponding to solutions that do not meet the design constraints. In a complex spatial deployment problem with real-time scheduling constraints, resource, co-location, and other constraints, randomly generating a set of initial solutions is unlikely to generate a set of valid initial solutions that satisfy the constraints (e.g., $co(\vec{T}) + p(\vec{T}) + r(\vec{T}) + s(\vec{T}) = 0$) [17]. Moreover, evolving solutions through arbitrary evolutionary schemes, such as genetic mating [17], is also unlikely to yield new valid solutions where the constraints are satisfied.

One approach to handling this issue is to devise a scoring solution that always ranks valid solutions higher than invalid solutions, but also scores invalid solutions on their relative *closeness* to being correct. For example, an invalid solution can be scored based on the number of constraints it violates:

$$-1 * (co(\vec{T}) + p(\vec{T}) + r(\vec{T}) + s(\vec{T}))$$

. The challenge with this strategy is that the number of violated constraints is not necessarily an accurate predictor of the solution's distance from a correct solution. For instance, a solution with a single violated constraint, may require changing every component to hardware allocation in order to meet the constraints. If the solution is not close to valid, the design may be promoted in the evolutionary process due to its good score, even though it is not remotely close to a good solution.

Another remedy is to devise a repair operation that can take an arbitrary invalid solution and modify it to yield a valid solution. With complex spatial deployment problems, however, the constraints are too complex to determine how to modify an arbitrary invalid solution to satisfy the deployment constraints.

3.3 ScatterD Hybrid Evolutionary Algorithm Modifications

To overcome these challenges of applying evolutionary algorithms to spatial deployment problems, ScatterD is based on a combination of first-fit bin-packing and an evolutionary algorithm. The evolutionary algorithm that is used with ScatterD is pluggable. We demonstrate the use of ScatterD with both PSO and a genetic deployment algorithm below.

In these evolutionary algorithms, a population of candidate deployment topologies, P, is randomly generated and then evolved over a series of time steps. At each time step, the population members are scored using a fitness measurement, $F(\vec{p_i})$. An evolutionary operator, $evolve(\vec{p_i})$, is then applied to each solution to either modify it before the next time step, remove it from the search process, or combine it with another solution to produce a new population memeber. The evolutionary operator's output is highly dependent on the scores received by the individual candidate designs.

In general, an evolutionary algorithm operates as follows:

1. Each member of the population in the evolutionary search process is assigned a random initial vector, $\vec{p_i} = \vec{random}$. In a genetic algorithm, the vector represents the genes in the candidate design. In a PSO algorithm, the vector specifies the position of a search particle in the multidimensional search space.

2. Each population member position is scored using a fitness metric, $F(\vec{p_i})$.

3. An evolutionary operator, $evolve(\vec{p_i})$, is applied to each population member to produce the population members for the next iteration of the algorithm.

4. Steps 2-3 are repeated until the maximum number of steps is reached or the process converges on a single solution.

5. The highest scoring population member is output as the result.

3.4 Combining Bin-packing with an Evolutionary Algorithm

To overcome the challenges of applying evolutionary algorithms to spatial deployment problems, ScatterD is based on a combination of first-fit bin-packing and an evolutionary algorithm. The most straightforward method of applying an evolutionary algorithm to spatial deployment is to model each population member's vector as a deployment topology, $\vec{T} = \vec{p_i}$. The fitness function for the evolutionary optimization can then be set to the power consumption calculation function, $F(\vec{p_i}) = e(\vec{T})$. Since the scheduling and other deployment constraints must also be upheld, the fitness function can refined and defined as:

$$V(\vec{p_i}) = co(\vec{p_i}) + p(\vec{p_i}) + r(\vec{p_i}) + s(\vec{p_i})$$

$$F(\vec{p_i}) = \begin{cases} e(\vec{p_i}) & \text{if } V(\vec{p_i}) = 0, \\ -1 * V(\vec{p_i}) & \text{otherwise.} \end{cases} \quad (4)$$

Since the spatial deployment solution space is highly constrained, using an arbitrary evolutionary operator, $evolve(\vec{p_i})$, to update p_i is likely to produce a large number of solutions for which the constraints do not hold and hence $F(\vec{p_i}) = -1 * V(\vec{p_i})$. Moreover, randomly assigning initial vectors, $\vec{p_i} = \vec{random}$, is unlikely to yield a valid deployment topology. The evolution may therefore progress using potentially highly inaccurate estimates of how close the solution is to yielding a spatial deployment topology that is valid and minimizes power consumption. Section 3 presents empirical results for a number of deployment problems that show this inaccuracy in solution quality estimation is a significant detriment to the spatial deployment topology search process.

To minimize the probability that invalid spatial deployment topologies are explored by the evolutionary optimization, ScatterD uses several specialized techniques to overcome these limitations. The two key challenges that ScatterD must use these techniques to address are (1) determining how to generate the initial vectors to maximize the probability that they correspond to valid deployment topologies and (2) ensuring that as the vectors are are evolved, the probability that they are invalid is minimized.

ScatterD combines an evolutionary algorithm with a bin-packing algorithm to address these two issues. The key change made to the evolutionary algorithm is that the population member's vectors are not interpreted as directly representing a deployment topology $\vec{T}$ but instead represent a bin-packing order for a subset of the components. This intermediate input is input into a bin-packing algorithm to produce the actual deployment topology, $\vec{T}$.

The hypothesis is that since a bin-packer uses specialized heuristic knowledge to deploy components, it is more likely to derive a deployment topology that is valid, compared with a randomized evolutionary scheme. A bin-packer, however, is designed to produce a single solution, which as shown in Section 2.2 is not necessarily optimized for power consumption minimization. The semi-random packing vector, produced by the evolutionary algorithm, serves to vary the constraints in the bin-packing problem and coax the heuristic bin-packing algorithm into producing multiple solutions. The evolutionary process thus becomes a driver for exploring the solution space via semi-randomized executions of a bin-packing algorithm.

3.5 Packing Order Vectors

In a standard heuristic bin-packer, such as first-fit bin-packing or best-fit bin-packing, the algorithm sorts the software components according to a heuristic, such as highest CPU consumption. In ScatterD's modified scheme, the bin-packer first places a semi-random subset of the components using the ordering obtained by interpreting the vector of a population member, $\vec{p_i}$, from the evolutionary algorithm as a packing order for a subset of the components. Only after those components are placed is the bin-packer allowed to sort and place the remaining components using its standard heuristics.

The key step in combining the two algorithms is producing a modified bin-packing algorithm that packs software components onto hardware nodes in two phases. In the first phase, the packing order vector is input into the bin-packing algorithm and the bin-packer iteratively removes the component at the head of the list, finds the first hardware node that will support it, and deploys it there. The packing order vector covers a random subset of the components that must be deployed. Any components packed in the first stage are removed from the master list of components to pack.

In the second phase, the bin-packing algorithm applies its standard heuristics to sort the remaining components. After the components are sorted, the bin-packer iteratively chooses the item at the head of the list of remaining items to pack and places it on the first valid hardware node placement. A placement on a specific hardware node is considered valid if it does not produce any constraint violations, *e.g.*, it does not cause $V(\vec{p_i} > 0$. This process proceeds until all of the components are placed on nodes.

The modified evolutionary algorithm works as follows with the hybrid modifications:

1. Each population member in the evolutionary search process is assigned a random initial vector, $\vec{p_i} = \vec{random}$.

2. For $i = 0$, $i < |p_i|$, a first-fit bin-packing algorithm

takes the software component referred to by position i and places it on a hardware node. The node that each component is placed on is recorded in the deployment topology vector, $T = dp_i$.

3. The software components that are not placed on a node in Step 2 are placed into a list, R.

4. The software components in R are sorted using a bin-packing heuristic, such as CPU consumption.

5. Each software component in R is placed on a hardware node using a standard bin-packing algorithm. The node that each component is placed on is recorded in the deployment topology vector, dp_i.

6. The score for each population member is calculated using a fitness metric, as a function of the deployment plan, $F(\vec{dp_i})$, and not directly from the population member's vector, $\vec{p_i}$.

7. An evolutionary operator, $evolve(\vec{p_i})$, is applied to each population member to produce the population members for the next iteration of the algorithm.

8. Steps 2-7 are repeated until the maximum number of steps is reached or the process converges on a single solution.

9. The highest scoring deployment topology, dp_i, is returned as the result.

3.6 Example: Using ScatterD to Create a Hybrid Heuristic Particle Swarm Optimization

PSO is an evolutionary algorithm designed to simulate the flocking behavior of birds searching for food. A virtual flock of birds, called particles, is created and each particle is scored using a fitness metric, $F(\vec{p_i})$. Particles are attracted to the best scoring position and attempt to move towards it. Each particle also remembers the best solution it has seen so far and uses this position in determining its movement. As the particles move, new global and personal bests are discovered, altering the flocking behavior of the particles.

In general, the PSO algorithm operates as follows:

1. Each member of the population in the evolutionary search process is assigned a random initial position, $\vec{p_i} = \vec{random}$.

2. Each population member position is scored using a fitness metric, $F(\vec{p_i})$.

3. The particle position, p_k, with the overall best score is marked as the global best, $\vec{G} = \vec{p_k}$.

4. Each particle has its score compared to the particle's personal best score obtained so far. If the score is higher, the personal best position, $\vec{pb_i}$, is set to the current position, $\vec{pb_i} = \vec{p_i}$.

5. The position of each particle, $p_i \in P$, is used to calculate a velocity for the particle:

$$\vec{v_i} = \vec{v_i} + R_1 C_1(\vec{p_i} - \vec{pb_i}) + R_2 C_2(\vec{p_i} - \vec{G})$$

The coefficients R_1 and R_2 are random numbers between zero and one. The coefficients $C1$ and $C2$ are used to alter the influence of the global and personal best particle positions on the velocity calculation.

6. Each particle's position for the next iteration of the evolution is updated by adding its new velocity, $\vec{p_i} = \vec{p_i} + \vec{v_i}$.

7. Steps 2-5 are repeated until the maximum number of steps is reached or the process converges on a single solution.

8. The highest scoring particle position is output as the result.

Modifying PSO to operate with the ScatterD is straightforward. The scoring function is modified to interpret the particle positions as packing order vectors for a bin-packer. The modified PSO algorithm works as follows:

1. Each particle in the evolutionary search process is assigned a random initial position, $\vec{p_i} = \vec{random}$.

2. For $i = 0$, $i < |p_i|$, a first-fit bin-packing algorithm takes the software component referred to by position i and places it on a hardware node. The node that each component is placed on is recorded in the deployment topology vector, $T = dp_i$.

3. The software components that are not placed on a node in Step 2 are placed into a list, R.

4. The software components in R are sorted using a bin-packing heuristic, such as CPU consumption.

5. Each software component in R is placed on a hardware node using a standard bin-packing algorithm. The node that each component is placed on is recorded in the deployment topology vector, dp_i.

6. The score for each particle's position is calculated using a fitness metric as a function of the deployment plan, $F(\vec{dp_i})$, and not the particle position, $\vec{p_i}$.

7. The particle position with the overall highest scoring deployment plan is marked as the global best, $\vec{G} = \vec{p_i}$. The global best position is *not* set to the deployment topology vector of that node.

8. Each particle has its deployment plan's score compared to the particle's personal best score obtained so far. If the score is higher, the personal best position, $\vec{pb_i}$, is set to the current position, $\vec{pb_i} = \vec{p_i}$.

9. The position of each particle, $\vec{p_i} \in P$, is used to calculate a velocity for the particle:

$$\vec{v_i} = \vec{v_i} + R_1 C_1(\vec{p_i} - \vec{pb_i}) + R_2 C_2(\vec{p_i} - \vec{G})$$

10. Each particle's position for the next iteration of the evolution is updated by adding its new velocity, $\vec{p_i} = \vec{p_i} + \vec{v_i}$.

11. Steps 2-10 are repeated until the maximum number of steps is reached or the process converges on a single solution.

12. The highest scoring deployment plan, dp_i, is output as the result.

Heuristic algorithms, such as bin-packing, are typically focused on finding exactly one solution to a problem. In the case of spatial deployment, this characteristic prevents the algorithms from exploring many potential designs that may provide excellent power consumption characteristics. Evolutionary algorithms, in contrast, lack the specialized problem-specific heuristic knowledge of an algorithm, such as bin-packing, but are excellent at exploring a number of design alternatives.

By combining the two approaches into a single algorithm, ScatterD leverages the key advantages of each alogirhtmic approach. ScatterD's soution space representation as bin-packing algorithm input permutations helps to decrease the probability that invalid solutions will be explored through arbitrary evolutionary population member permutations. The use of an evolutionary exploration technique also allows the evaluation of multiple candidate solutions and comparisons based on power consumption characteristics that are missed with a standard heuristic bin-packer.

4 Empirical Results

This section analyzes empirical results obtained by applying the ScatterD techniques, bin-packing, PSO, and a genetic algorithm to an example fractionated spacecraft system. The fractionated spacecraft example was produced by modifying a production avionics dataset [19] obtained from Lockheed Martin Aeronautics through the SPRUCE project (`www.sprucecommunity.org`), which is a web-accessible portal that pairs academic researchers with industry challenge problems complete with representative project data. We compared the ability of each deployment algorithm to reduce the power consumption provided by the baseline spatial deployment topology from the aeronautics dataset.

4.1 Experimental Platform

The fractionated spacecraft example contains a spatial deployment topology and software architectural information for a spacecraft system with ∼50 processors, ∼300 software components, and 15,000 periodic messages. Real-time scheduling constraints are included in the example, including task rates and CPU consumption. The example also contains information on the messaging interactions between the software components. A total of ∼15,000 messaging interactions are described by their rate, message size, message type, and source/destination software components.

We obtained the data in the following experiments by augmenting the original fractionated spacecraft example with power consumption information for the processors and network communications. The software architecture remained static throughout the experiments while we varied the power consumed by the processors to simulate different deployment scenarios with hardware varying from workstation Xenon processors to wireless sensor motes. Power dissipation of the various processors was based on public power dissipation data published by Intel and empirical results from sensor motes published by Anastasi et al. [2]. All network communications were modeled to take place over a wireless 802.11b network and power consumption was calculated using the results produced by Feeney et al. [20].

For each experiment we compared the deployments produced by six different automated deployment techniques against the baseline deployment from the aeronautics dataset. The six algorithm variations that we compared were:

1. **B**in-packing – A first-fit heuristic bin-packer was chosen due to their widespread use for deployment problems.

2. **S**catterD PSO – The PSO variant of ScatterD.

3. **S**catterD Genetic – The genetic algorithm variant of ScatterD.

4. **P**SO with Bin-packing Seeding – A PSO algorithm was seeded with the results of randomized first-fit bin-packing. Packing-order vectors were not utilized in this variant.

5. **G**enetic with Bin-packing Seeding – A genetic algorithm was seeded with the results of randomized first-fit bin-packing. Packing-order vectors were not utilized in this variant.

6. **P**SO – the same PSO algorithm used for the ScatterD PSO was used but without the packing-order vectors and initial seeding from a bin-packing algorithm.

7. **G**enetic Algorithm – the same genetic algorithm used for the ScatterD was used, but without the packing-order vectors and initial seeding from a bin-packing algorithm.

Variations of PSO and genetic algorithms with and without the ScatterD enhancements were chosen to compare the improvements yielded by the ScatterD techniques. A first-fit bin-packing algorithm was chosen because they are widely used in deployment problems.

For both the genetic and PSO algorithms, a population size of 20 was used and a total of 20 search iterations (generations) were run. The PSO algorithm used a local learning coefficient of 1 and a global learning coefficient of 2. The genetic algorithm allowed a total of 10% of the population to be passed through to the next generation, selected the top 25% of solutions for mating, and applied a mutation probability of 5%.

Each algorithm was implemented in Java (the implementations are available in open-source form through the Ascent Design Studio at `code.google.com/p/ascent-design-studio`). The comparisons were performed on an Apple MacBook Pro with 4gb of RAM, a 3.06ghz Intel Core 2 Duo Processor, and Mac OS X 10.5.7. A version 1.6 Java Virtual Machine (JVM) was used to conduct the experiments. The JVM was run in client mode using a heap size of 40 megabytes (-Xms40m) and a maximum memory size of 256 megabytes (-Xmx256m).

4.2 Experiment 1: Deployment Across a Homogeneous Set of Workstation Processors

This experiment tested the capabilities of each algorithm to reduce the power consumption of the avionics software in a deployment across a homogeneous set of 6 core Intel Xenon L7455 processors. Power consumption improvements were calculated by comparing the power consumption of the deployments produced by each algorithm to the original baseline deployment from the aeronautics dataset.

Hypothesis: ScatterD should provide significant reductions in power consumption compared to bin-packing. Initially, we believe that ScatterD's ability to search multiple spatial deployment permutations would lead to solutions with far superior power consumption characteristics than those produced by the bin-packing algorithm. We also hypothesized that neither the standard PSO or genetic algorithms would produce good solutions due to the complexity of the design space.

Analysis of results. The results from the experiment are shown in Figure 5. The ScatterD Genetic algorithm produced the largest power consumption reduction, followed closely by ScatterD PSO. Interestingly, the bin-packing algorithm produced a power savings that was within ≈94% of the solution produced by ScatterD Genetic.

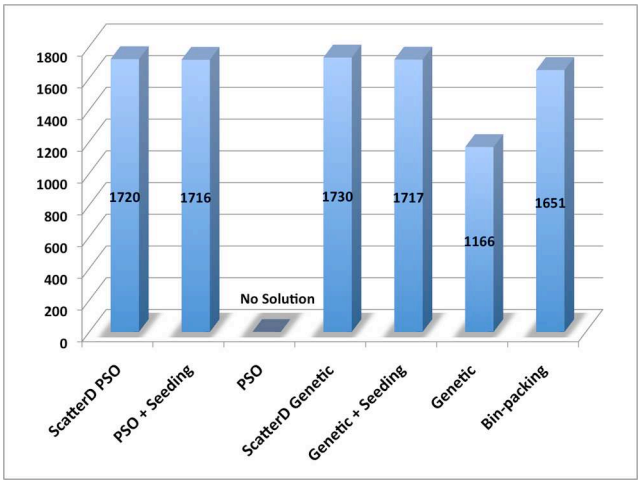


Figure 5. Homogeneous Xenon Deployment

After careful analysis, it became clear that because the processors are homogenous, the first-fit bin-packing heuristic performs extremely well. Moreover, because the Xenon processor draw a substantial amount of power compared to the network transmissions, idling processors was the most important concern in this problem. In this particular case, the emphasis on processor idling and the homogeneity of the cores allowed the bin-packing algorithm to nearly equal the more complex searches from ScatterD.

In this experiment, the standard PSO algorithm was unable to find a valid solution within the allotted time. The standard genetic algorithm also produced a poor solution. Throughout the experiments, both of these algorithms produced poor results compared to the rest, demonstrating the importance of ScatterD's hybrid heuristic optimizations.

4.3 Experiment 2: Deployment Across a Homogeneous Set of Low-power Processors

This experiment compared deployments across a group of Intel XScale processors, which consume substantially less power than the Xenon workstation processors. The lower power consumption of the processors meant that reductions in network traffic would play a more important role in reducing the power consumption in this scenario.

Hypothesis: An emphasis on network traffic reduction would produce far more power conserving deployment topologies from ScatterD versus bin-packing. Since the cost of CPU time was decreased, we assumed that ScatterD's advantages in network traffic optimization would allow it to produce much better solutions than bin-packing.

Analysis of results. The results for this experiment are shown in Figure 6. The ScatterD Genetic algorithm produced the biggest reduction in power consumption. In this case, the ScatterD Genetic algorithm saved roughly 25%

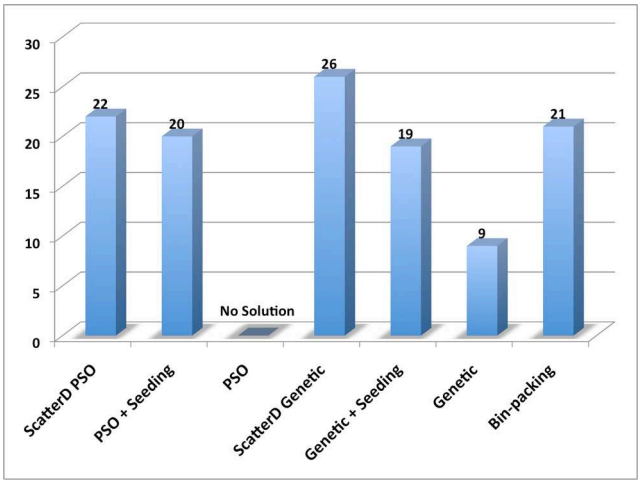


Figure 6. XScale Deployment

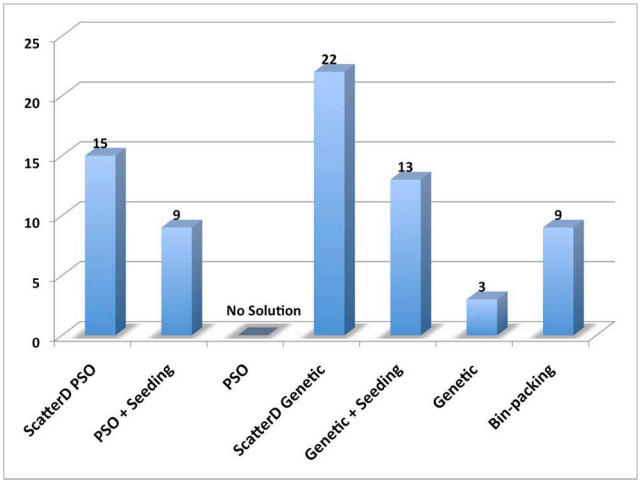


Figure 7. Sensor Motes Deployment

more power than the first-fit bin-packing algorithm. The ScatterD PSO algorithm also beat the bin-packing algorithm by roughly 3%.

The Genetic and PSO algorithms that did not use the full set of ScatterD optimizations were unable to produce a solution better than the bin-packing algorithm. The PSO algorithm did not find a solution in the allotted search time frame. Moreover, the genetic algorithm produce a solution that saved less than 50% of the power of the first-fit bin-packing algorithm's solution. In this scenario, the packing-order vector and bin-packing seeding optimizations of ScatterD produced a clear advantage over standard PSO and genetic algorithms.

4.4 Experiment 3: Deployment Across a Homogeneous Set of Sensor Motes

This experiment compared the deployment topologies produced by the algorithms when deploying the software onto a set of low power wireless sensor motes. The power consumption of the motes was modeled after the results produced by Anastasi et al. [2]. The extremely low power consumption of the processors meant that reductions in network traffic would play a far larger role in reducing the power consumption than in other scenarios.

Hypothesis: Network traffic reduction would be the dominant factor in determining which algorithm produced the lowest power spatial deployment topology. The significantly lower power consumption of CPU processing relative to network power consumption should favor algorithms that reduce network traffic the most. We anticipated that ScatterD would show further improvements in power consumption versus the previous two experiments.

Analysis of results. The results for this experiment are shown in Figure 7. The ScatterD Genetic algorithm produced the biggest reduction in power consumption. Moreover, the ScatterD Genetic's deployment topology saved 240% more power than the bin-packing algorithm. The ScatterD PSO algorithm also saved 167% more power than the bin-packing algorithm. The standard bin-packing seeded PSO and genetic algorithms also tied or surpassed the bin-packing algorithm. The standard genetic algorithm did not produce a power conservative deployment topology. The PSO algorithm did not find a valid solution.

As hypothesized, ScatterD significantly outperformed the standard bin-packing algorithm since takes into account network power consumption variations. Moreover, without the ScatterD optimizations, the standard genetic and PSO algorithms had difficulty in the highly constrained deployment search spaces. For deployments across low-power wireless sensor networks or in other low-power scenarios, ScatterD produces the biggest reductions in power consumption.

4.5 Experiment 4: Deployment Across a Heterogeneous Set of High and Low Power Processors

This experiment combined the power consumption models from the previous three experiment into a single deployment scenario across a set of heterogeneous processors. This scenario was designed to mimic a large-scale system combining a variety of servers, mobile devices, and wireless sensors. One third of the processors were modeled as high-powered Xenons. One third of the processors were modeled as lower powered XScale processors. Finally, the remaining processors were modeled as wireless sensor motes.

Hypothesis: Accounting for power consumption variations in processor cores would be the most significant factor in reducing power consumption. With a variety of

processor power consumption profiles, we believed that accounting for these variations would be more important than reducing network traffic. Again, we anticipated that ScatterD would outperform the other algorithms.

Analysis of results. The results for this experiment are shown in Figure 8. The ScatterD Genetic algorithm

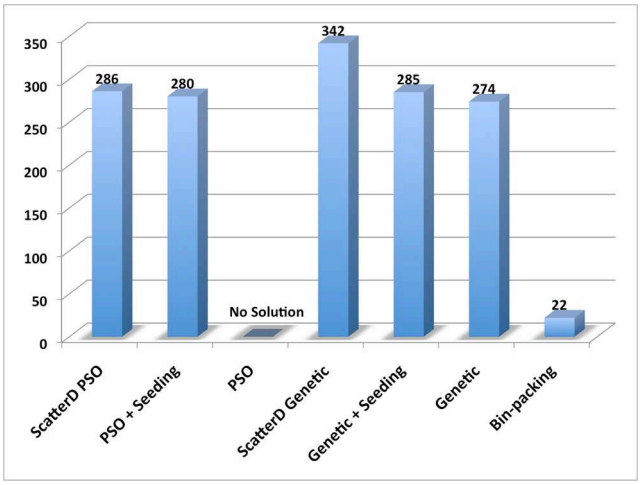


Figure 8. Heterogeneous XScale and Xenon Deployment

again produced the biggest reduction in power consumption. Suprisingly, the reduction in power consumption was over 1500% greater than the bin-packing algorithm. The Genetic PSO algorithm also exceeded the power reduction of the bin-packing algorithm by over 1300%.

While analyzing the results, we derived a simple modification to the problem that provided a substantial improvement for the bin-packing algorithm. Instead of providing the processors to the algorithms in an arbitrary order, we first sorted the list so that the lowest power processors were in the front of the list. The intuition is that a first-fit bin-packing algorithm will use these lowest power processor first, before moving on to more power consumptive processors. We also found that this simple modification yielded improvements from the other techniques as well.

The results of applying this modification to the problem and re-running it are shown in Figure 9. The modifications approximately doubled the power reductions produced by the bin-packing algorithm. The ScatterD PSO algorithm provided roughly 100% more power consumption savings compared to the bin-packing algorithm. The ScatterD Genetic algorithm also produced the best overall power consumption score.

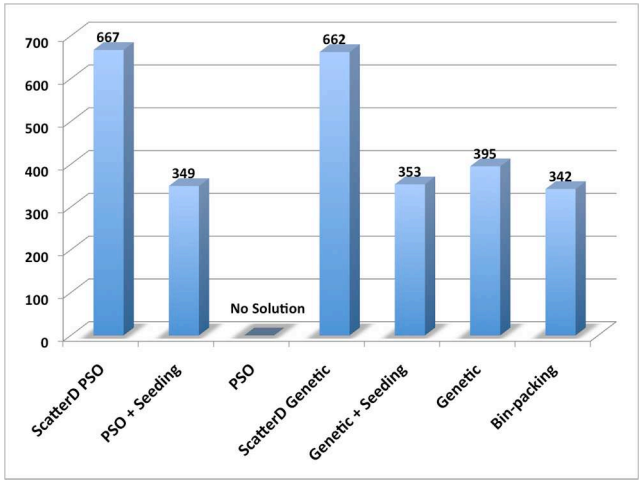


Figure 9. Heterogeneous XScale and Xenon Deployment with Bin-packing Sorting Optimization

4.6 Experiment 5: Measuring Power Consumption Improvements from Reduced Network Traffic

This experiment measured the power consumption improvements yielded by the algorithms through reduced network traffic. Processor time was modeled as consuming zero power to allow us to only measure variations in power consumption from network traffic reductions.

Hypothesis: Significant improvements in power consumption can be achieved by co-locating software components with high traffic interactions. Although reducing the number of processors in a spatial deployment topology can produce significant power consumpton improvements, we believed that network traffic reduction could also play an important role. Bin-packing would provide some reduction in network traffic by packing software more tightly onto the processors and co-locating more components as a side-effect. ScatterD should be able to outperform bin-packing with its more complex search techniques.

Analysis of results. The results for this experiment are shown in Figure 10. The ScatterD PSO algorithm produced the biggest reduction in power consumption. The ScatterD PSO algorithm saved 70% more power than the bin-packing algorithm. The ScatterD Genetic algorithm reduced power consumption by 60% more than the bin-packing algorithm. The standard variants of PSO and genetic deployment algorithms did not perform as well as bin-packing.

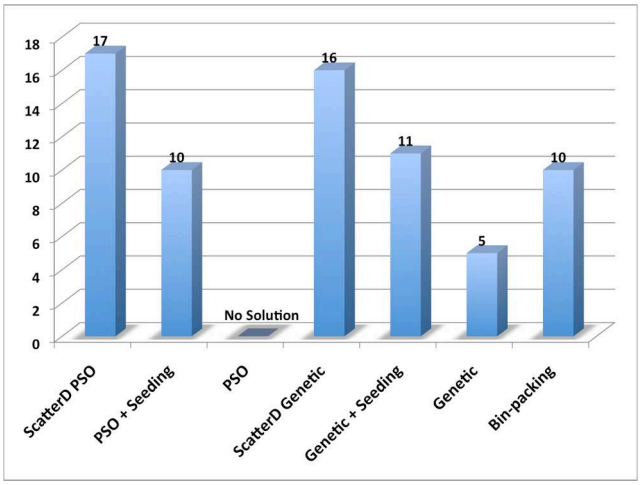


Figure 10. Deployment with Wireless Network Only Power Consumption Optimization

5 Related Work

A number of prior research efforts are related to the spatial deployment problem presented in this paper. This section provides a taxonomy of these related works and compares and contrasts them to ScatterD. The related works are categorized based on the type of algorithm used in the deployment process.

Multi-processor scheduling. Bin-packing algorithms have been successfully applied to the NP-Hard problem of multi-processor scheduling [9]. Multi-processor scheduling requires finding an assignment of real-time software tasks to hardware processors, such that no tasks miss any deadlines. A number of bin-packing modifications are used to optimize the assignment of the tasks to use as few processors as possible [16, 14, 35, 15, 9]. The chief issue of using these existing bin-packing algorithms for spatial deployment optimization to minimize power is that they focus on minimizing total processors used.

In certain specialized scenarios, such as when all processors are identical and network communication is not a major power concern, these algorithms are suitable for performing power consumption optimization. If processors vary in performance per watt or power consumptive wireless communication is involved, however, these algorithms will not make ideal placement decisions. In contrast, ScatterD, is designed to account for these variations in power consumption and make appropriate placement decisions.

Kirovski et al. [31] have developed heuristic techniques for assigning tasks to processors in resource constrained systems to minimize power consumption. Their technique optimizes a combination of variations in processor power consumption and voltage scaling. These techniques, however, do not account for network communication in the power optimization process.

Okuma and others [38, 1] have developed scheduling techniques for real-time task scheduling on variable voltage processor cores. The scheduling algorithm both optimizes task ordering and voltage levels to minimize power consumption. Changes in processing time from lowering processor voltage are also considered in the scheduling process. Others, such as Aydin et al. [3, 25], have also investigated real-time task scheduling but across multiple processors [50]. These techniques, however, do not consider how network power consumption impacts the overall system.

Task graph scheduling to minimize energy consumption in embedded systems. Energy consumption optimization has also been investigated in the context of the scheduling of conditional and unconditional task graphs to underlying hardware processors [43, 41]. This work attempts to optimize power consumption while ensuring that tasks execute on processors in a required order and meet minimum performance requirements. In general, however, this work has focused on embedded systems that are spatially co-located and does not account for network usage energy consumption.

Power optimization for multicore systems on a chip. A large amount of research has focused on power consumption minimization in multicore systems on a chip [28, 34, 40, 38, 23, 7]. These research efforts have investigated methods of leveraging differing voltages and capabilities of specific processing cores to optimize the power consumption of the system while guaranteeing performance constraints. This work mainly focuses on the core assignment aspects of the allocating computing cores to software tasks.

Power optimization for network systems on a chip. Others have considered network power consumption in the context of network systems on a chip, which allow for variable speed interconnect links [26, 44]. These techniques optimize task placement and network link speed in the context of a single chip. Spatially-separated systems on different chips are not considered in this work.

Hardware/software co-synthesis. Hardware/Software co-synthesis research has yielded techniques for determining the number of processing units, task scheduling, and other parameters to optimize systems for power consumption while meeting hard real-time constraints. Dick et al. [17, 18], have used a genetic algorithm for the co-synthesis problem. As with other single-chip work, however, this research is directed towards systems that are not spatially separated from one another.

Client/Server Task Partitioning for Power Optimization. Network power consumption and processor power consumption have both been considered in work on partitioning client/server tasks for mobile computing [49, 12, 36]. In this research, the goal is to determine how to parti-

tion tasks between a server and mobile device to minimize power drain on the device. This work, however, is focused only on how power is saved by moving processing responsibilities between a single client and server.

6 Concluding Remarks and Lessons Learned

Power consumption in DRE systems can be minimized by carefully choosing a good spatial deployment topology. The spatial deployment topology determines how far apart communicating software components are and thus how much transmit power is needed for wireless communication. Moreover, processing nodes vary in performance per watt for different tasks and these variations can be exploited to optimize the energy consumption footprint of the system.

Designing automated algorithms to perform deployment optimization is hard, *e.g.*, deployment with real-time scheduling constraints is an NP-Hard problem [9]. Exponential algorithms, such as integer programming, often do not scale well for industrial size problems. Evolutionary algorithmic techniques, such as genetic algorithms, tend to provide poor performance in the highly-constrained solution spaces that are typical of spatial deployment problems.

This paper described how ScatterD combines heuristic bin-packing with an evolutionary algorithm to obtain substantial performance increases in spatial deployment topology derivation. In the empirical results presented in Section 4, we observed substantial power consumption improvements with the hybrid ScatterD algorithm compared to strictly evolutionary techniques. Moreover, ScatterD produced deployment topologies that saved 4.5% more power in server deployments (Experiment 1, Section 4.2) and saved over 240% more power in wireless sensor mote deployment scenarios (Experiment 3, Section 4.4).

We learned the following lessons from our work researching and developing the ScatterD hybrid heuristic/evolutionary deployment algorithm:

- **Power consumption can be reduced significantly by carefully balancing network utilization and processor efficiency tradeoffs.** The empirical data from Section 4 showed that accounting for both network and processor power consumption provides superior results to focusing exclusively on either one individually.

- **Solution space representation is critical** to the ability of an evolutionary algorithm to efficiently generate spatial deployment topologies. By modeling the solution space as permutations of the input to a heuristic bin-packing algorithm, we observed a roughly four-fold increase in the number of good deployment topologies that were explored.

- **Combining heuristic and evolutionary algorithms over comes a number of problems** of applying either independently to spatial deployment topology derivation. For example, by combining first-fit bin-packing with PSO, a hybrid algorithm can be created that has a lower probability of generating invalid solutions during the evolutionary solution permutation process.

- **Determining the appropriate method of integrating heuristic and evolutionary algorithms is not easy**, significant experimentation is required to find a way of integrating the two techniques that yields good results. For example, preliminary experiments that we performed that used evolutionary techniques to adjust the relative weights of bin-packing items did not perform nearly as well as modeling the solution space as packing-order vectors.

- **Other combinations of heuristic and evolutionary algorithms may provide superior results.** Each heuristic algorithm has unique problem characteristics for which it produces excellent results. In future work, therefore, we plan to investigate other combinations of heuristic and evolutionary algorithms that perform well on spatial deployment optimization problems.

An implementation of the ScatterD deployment algorithm is available in open-source form as part of the Ascent Design Studio (`code.google.com/p/ascent-design-studio`).

References

[1] T. AlEnawy and H. Aydin. Energy-aware task allocation for rate monotonic scheduling. In *Proceedings of the 11th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS05)*, pages 213–223, 2005.

[2] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori. Performance measurements of motes sensor networks. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 174–181. ACM New York, NY, USA, 2004.

[3] H. Aydin and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. In *Proceedings of 17th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 113–121, 2003.

[4] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, USA, 1996.

[5] M. Bastarrica, A. Shvartsman, and S. Demurjian. A Binary Integer Programming Model for Optimal Object Distribution. In *2nd Int. Conf. on Principles of Distributed Systems*.

[6] H. Beitollahi and G. Deconinck. Fault-Tolerant Partitioning Scheduling Algorithms in Real-Time Multiprocessor

Systems. *Pacific Rim International Symposium on Dependable Computing, IEEE*, 0:296–304, 2006.

[7] L. Benini, D. Bertozzi, A. Guerri, and M. Milano. Allocation, scheduling and voltage scaling on energy aware mpsocs. *Lecture Notes in Computer Science*, 3990:44, 2006.

[8] O. Brown, P. Eremenko, and B. Hamilton. The value proposition for fractionated space architectures. *Sciences*, 99(1):2538–2545, 2002.

[9] A. Burchard, J. Liebeherr, Y. Oh, and S. Son. New Strategies for Assigning Real-time Tasks to Multiprocessor Systems. *IEEE Transactions on Computers*, 44(12):1429–1442, 1995.

[10] A. Carzaniga, A. Fuggetta, S. Richard, D. Heimbigner, A. van der Hoek, A. Wolf, and COLORADO STATE UNIV FORT COLLINS DEPT OF COMPUTER SCIENCE. *A Characterization Framework for Software Deployment Technologies*. Defense Technical Information Center, 1998.

[11] K. Chakrabarty, S. Iyengar, H. Qi, and E. Cho. Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks. *IEEE Transactions on Computers*, pages 1448–1453, 2002.

[12] G. Chen, B. Kang, M. Kandemir, N. Vijaykrishnan, M. Irwin, and R. Chandramouli. Studying energy trade offs in offloading computation/compilation in Java-enabled mobile devices. *IEEE Transactions on Parallel and Distributed Systems*, pages 795–809, 2004.

[13] W. Damm, A. Votintseva, A. Metzner, B. Josko, T. Peikenkamp, and E. Böde. Boosting Re-use of Embedded Automotive Applications Through Rich Components. *Proceedings of Foundations of Interface Technologies*, 2005, 2005.

[14] S. Davari and S. Dhall. An On-line Algorithm for Real-time Tasks Allocation. In *IEEE Real-time Systems Symposium*, pages 194–200, 1986.

[15] S. Davari and S. Dhall. On a Periodic Real-Time Task Allocation Problem. In *19th Annual International Conference on System Sciences*, pages 133–141, 1986.

[16] S. Dhall and C. Liu. On a Real-time Scheduling Problem. *Operations Research*, 26(1):127–140, 1978.

[17] R. Dick and N. Jha. MOGAC: A multiobjective genetic algorithm for the co-synthesis of hardware-software embedded systems. In *Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, pages 522–529. IEEE Computer Society Washington, DC, USA, 1997.

[18] R. Dick and N. Jha. MOCSYN: Multiobjective core-based single-chip system synthesis. In *Proceedings of the conference on Design, automation and test in Europe*. ACM New York, NY, USA, 1999.

[19] B. Dougherty, J. White, J. Balasubramanian, C. Thompson, and D. C. Schmidt. Deployment Automation with BLITZ. In *Emerging Results track at the 31st International Conference on Software Engineering*, Vancouver, CA, May 2009.

[20] L. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM*, volume 3, pages 1548–1557. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 2001.

[21] D. Fogel, N. Inc, and C. La Jolla. What is Evolutionary Computation? *Spectrum, IEEE*, 37(2):26–28, 2000.

[22] P. Hladik, H. Cambazard, A. Déplanche, and N. Jussien. Solving a real-time allocation problem with constraint programming. *The Journal of Systems & Software*, 81(1):132–149, 2008.

[23] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, M. Srivastava, S. Inc, and M. View. Power optimization of variable-voltage core-based systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(12):1702–1714, 1999.

[24] J. Hooker. Planning and scheduling by logic-based benders decomposition. *OPERATIONS RESEARCH-BALTIMORE THEN LINTHICUM-*, 55(3):588, 2007.

[25] H. Hsu, J. Chen, and T. Kuo. Multiprocessor synthesis for periodic hard real-time tasks under a given energy constraint. In *Proceedings of the conference on Design, automation and test in Europe: Proceedings*, pages 1061–1066. European Design and Automation Association 3001 Leuven, Belgium, Belgium, 2006.

[26] J. Hu and R. Marculescu. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Proceedings of the 2003 conference on Asia South Pacific design automation*, pages 233–239. ACM New York, NY, USA, 2003.

[27] X. Hu and R. Eberhart. Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization. In *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics*, volume 5, pages 203–206, 2002.

[28] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Proceedings of the 1998 international symposium on Low power electronics and design*, pages 197–202. ACM New York, NY, USA, 1998.

[29] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, 1995.

[30] J. Kinnebrew, N. Shankaran, G. Biswas, and D. Schmidt. A Decision-Theoretic Planner with Dynamic Component Reconguration for Distributed Real-Time Applications. In *Poster paper at the Twenty-First National Conference on Artificial Intelligence*, Boston, MA, July 2006.

[31] D. Kirovski and M. Potkonjak. System-level synthesis of low-power hard real-time systems. In *Proceedings of the 34th annual conference on Design automation*, pages 697–702. ACM New York, NY, USA, 1997.

[32] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[33] J. Koza and J. Rice. *Genetic Programming*. Springer, 1992.

[34] W. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. *ACM Transactions on Embedded Computing Systems (TECS)*, 4(1):211–230, 2005.

[35] S. Lauzac, R. Melhem, and D. Mosse. An efficient RMS Admission Control and its Application to Multiprocessor Scheduling. In *International Parallel Processing Symposium*, pages 511–518, 1998.

[36] Z. Li, C. Wang, and R. Xu. Task allocation for distributed multimedia processing on wirelesslynetworked handheld devices. In *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pages 79–84, 2002.

[37] A. Nechypurenko, E. Wuchner, J. White, and D. C. Schmidt. Application of Aspect-based Modeling and Weaving for Complexity Reduction in Development of Automotive Distributed Realtime Embedded Systems. In *Proceedings to the Sixth International Conference on Aspect-Oriented Software Development*, Vancouver, British Columbia, Mar. 2007.

[38] T. Okuma, T. Ishihara, and H. Yasuura. Real-time task scheduling for a variable voltage processor. In *Proc. International Symposium on System Synthesis*, pages 24–29, 1999.

[39] B. Powell and A. Perkins. Fleet Deployment Optimization for Liner Shipping: An Integer Programming Model. *Maritime Policy & Management*, 24(2):183–192, 1997.

[40] G. Quan and X. Hu. Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. In *Proceedings of the 38th conference on Design automation*, pages 828–833. ACM New York, NY, USA, 2001.

[41] D. Roychowdhury, I. Koren, C. Krishna, and Y. HL. A voltage scheduling heuristic for real-time task graphs. In *Dependable Systems and Networks, 2003. Proceedings. 2003 International Conference on*, pages 741–750, 2003.

[42] Y. Shi and R. Eberhart. Empirical Study of Particle Swarm Optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 1948–1950. Piscataway, NJ: IEEE Service Center, 1999.

[43] D. Shin and J. Kim. Power-aware scheduling of conditional task graphs in real-time multiprocessor systems. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 408–413. ACM New York, NY, USA, 2003.

[44] D. Shin and J. Kim. Power-aware communication optimization for networks-on-chips with voltage scalable links. In *Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 170–175. ACM New York, NY, USA, 2004.

[45] R. Simmons, D. Apfelbaum, D. Fox, R. Goldman, K. Haigh, D. Musliner, M. Pelican, and S. Thrun. Coordinated Deployment of Multiple, Heterogeneous Robots. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 3, 2000.

[46] J. Stankovic. Strategic Directions in Real-time and Embedded Systems. *ACM Computing Surveys (CSUR)*, 28(4):751–763, 1996.

[47] N. R. C. Steering Committee for the Decadal Survey of Civil Aeronautics. *Decadal Survey of Civil Aeronautics: Foundation for the Future*. The National Academies Press, 2996.

[48] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40(2):117–134, 1994.

[49] C. Wang and Z. Li. A computation offloading scheme on handheld devices. *Journal of Parallel and Distributed Computing*, 64(6):740–746, 2004.

[50] C. Xian, Y. Lu, and Z. Li. Energy-aware scheduling for real-time multiprocessor systems with uncertain task execution time. In *Proceedings of the 44th annual conference on Design automation*, pages 664–669. ACM New York, NY, USA, 2007.